CS 224N FINAL PROJECT REPORT
REGBASE: An Information Infrastructure for Regulation Management and Analysis
Gloria T. Lau (glau@stanford.edu) Jun 8th, 2002

1. Introduction and Motivation to the Problem

Government regulations are an important asset of the society; ideally they should be readily available and retrievable by the general public. Curious citizens are entitled to and thus should be provided with the means to better *understand* government regulations. However, due to the heavy reference structure of regulations, an average citizen is likely to soon get lost in the jungle of definitions of legal terms and links. Multiple sources of regulations, e.g. from the federal or state government, or from local offices, further complicates the matter. These different sources of regulations sometimes compliment and modify each other, while they might as well contradict with one another. Therefore, to ease understanding of regulations by the general public, tools are needed to locate similar or related code sections, and to compare and contrast them if possible.

To further motivate the problem and also to understand why traditional vector model of text documents cannot be used as the sole algorithm for comparison and analysis, here is an example [4] of conflicting sections, with the first focuses on wheelchair traversal while the second focuses on the visually impaired when using a cane:

ADAAG 4.7.2
Slope. …Transitions from ramps to walks, gutters, or streets shall be flush and free of abrupt changes…
CBC 1127B.5.5
Beveled lip. The lower end of each curb ramp shall have a ½ inch (13mm) lip beveled at 45 degrees as a detectable way-finding edge for persons with visual impairments.

The ultimate goal here is to locate the conflicting idea captured by the word *flush* and the measurement *½ inch lip beveled at 45 degrees*. This is impossible without domain-specific information as well as general word sense comparison.

2. Literature Review and Related Work

Text document comparison, in particular similarity analysis between documents is widely studied in the field of information retrieval (IR). Techniques such as the Boolean model and the Vector model exist [2], and most of them are bag-of-word type of analysis. This type of models cannot capture synonymy information without the help of thesauri, and Latent Semantic Indexing (LSI) comes in to fill the gap between word and concept. LSI uses an algorithm called Singular Value Decomposition (SVD) to reduce the dimension of term space into concept space; the claim is that synonym groups or words that represent the same concept are mapped onto concept axes.

The regulations are heavily referenced, and link analysis is handy to provide a more reliable similarity measure. Academic citation analysis is closest in this regard; however the algorithm cannot be transported to our domain directly: citation analysis assumes a pool of documents citing one another; however our problem here is separate *islands* of information where *within* island documents are highly referenced; *across* island they aren't. Different algorithm is sought to serve the purpose.

In addition, bipartite graph matching problem, in particular the stable marriage algorithm is also briefly reviewed. Each document can be viewed as a graph, and our problem can be cast as matching nodes (sections of regulations) across two graphs. Future work is anticipated to further study the application of this model on our domain.

3. Scope and Approach

3.1 Prevous Work

In this project we work with two sets of data: ADAAG (Americans with Disabilities Act Accessibility Guide) and UFAS (Uniform Federal Accessibility Standards) [1], both of which are federal standards and are expected to be relatively similar. Previous work has been done to consolidate the data format, and currently both are in XML with sections separated. Since regulations are hierarchical, intuitively they are tree structured, e.g. Section 2.4.6 is a child of Section 2.4, and thus a section is represented as a node in a regulation tree and this hierarchy is also captured in the XML regulation structure. Machine-extracted domain-specific information, e.g. measurement tags, as well as key phrase concepts identified by a software called Semio, are integrated into the document as added XML tags.

3.2 Scope of this Project

Starting from the modified tree-structure corpus described above, our goal in this project is to develop a system to compare sections in different regulations; precisely, given two sections (a, u) each from a different regulation (ADAAG, UFAS), a pairwise initial similarity score between 0 and 1 is produced by comparison of tags. Then the immediate surrounding nodes, in particular the parent, the siblings and the children of (a, u) are compared as well to modify their initial score. The influence of the not-so-immediate surroundings of nodes a and u is taken into account by a process called Reference Distribution. Here, based on the assumption that similar sections reference similar sections, the referees' score is used to update that of (a, u). After this process, a stable ranking of the most similar or related sections is produced.

To assess the performance of this system, a control experiment will be implemented using LSI techniques. Each section is represented by a vector of words, or concepts if SVD is performed; pairwise comparison of sections can be obtained from the cosine similarity of vectors, or other similarity measures as discussed in Section 8.5 in the course textbook. Note that the *ranking*, not the actual similarity score, will be compared with that of our system. Our hope is that by utilizing the structure of regulations and the addition of domain-specific knowledge captured by the extra tags, our system will perform better than a bag-of-word type of comparison such as LSI; or at the very least, provide additional useful information in comparison and ranking.

4. Algorithm

Please also see the "Future Works" for further discussion of alternative algorithms investigated but not implemented for various reasons.

4.1 The Base Score: f_0

The framework in the implementation is set to perform 3 types of matching: concept match, measurement match and title match. However, due to press in time and the yet-to-be-solved scoring method for measurement tags, only concept match is performed to illustrate the idea.

A vector of concept frequencies is built for each section, with the length normalized to one. This is similar to a term frequency × inverse document frequency ($tf \times idf$) type of measure; however our system do not incorporate the *idf* component in concept analysis, since tagged concepts are already rare and we do not wish to reduce the similarity with the *idf* component. The distance between vectors are taken to be their cosine similarity which gives a f_0 between 0 and 1.

4.2 The Refined Scores

There is one central assumption in the refinement techniques: we are only interested in increasing the similarity but not reducing it. Thus in the following sections we only consider neighbors or referenced sections that already have higher similarity scores than the pair of interest. The validity of such an assumption will be discussed below in Section 4.4.

In addition, it is worth nothing that the refinement process is sequential: $f_{s\text{-}psc}$ is based on f_0, while $f_{psc\text{-}psc}$ is based on $f_{s\text{-}psc}$, and f_{rd} is based on $f_{psc\text{-}psc}$.

4.2.1 Self vs. Psc: $f_{s\text{-}psc}$

Based on f_0, the first refinement we can perform is to utilize the tree structure of regulations. The immediate surrounding of section A, namely the parent, the siblings and the children, collectively labeled set psc(A), can be compared against section B itself, and vice versa, to refine f_0(A,B). Again, section A and B are on different trees, and as explained above, we are only interested in s-psc scores higher than what A and B already has. We have

Set $S = f_0(\text{A}, \text{psc(B)}) \cup f_0(\text{psc(A)}, \text{B})$
N = sizeof(S)
$\delta_{GT} = \Sigma_{s>f0(\text{A,B})} (s - f_0(\text{A,B})),\ s \in S$
$\alpha_{s\text{-}psc}$ = discount factor of update
if (N != 0) $\quad f_{s\text{-}psc}(\text{A,B}) = f_0(\text{A,B}) + \alpha_{s\text{-}psc} \times (\delta_{GT} / N)$
else $\quad f_{s\text{-}psc}(\text{A,B}) = f_0(\text{A,B})$

Here, set S is the set of similarity scores of section A and psc(B), and psc(A) and section B. δ_{GT} sums over all of the s in set S that are greater than the original score; thus δ_{GT} / N represents the average greater-than score. Clearly α is always less than 1 so that self-self comparison always weighs more than self-psc comparison.

4.2.2 Psc vs. Psc: $f_{psc\text{-}psc}$

Based on $f_{s\text{-}psc}$, the second refinement is to account for the influence of psc-psc on section A and B. Here psc(A) is compared against psc(B) to refine f_0(A,B), so another layer of indirection is inferred and thus the weight of psc-psc should be less than that of s-psc. We have

Set $S = f_{psc\text{-}psc}(\text{psc(A)}, \text{psc(B)})$
N = sizeof(S)
$\delta_{GT} = \Sigma_{s>fs\text{-}psc(\text{A,B})} (s - f_{s\text{-}psc}(\text{A,B})),\ s \in S$
$\alpha_{psc\text{-}psc}$ = discount factor of update
if (N != 0) $\quad f_{psc\text{-}psc}(\text{A,B}) = f_{s\text{-}psc}(\text{A,B}) + \alpha_{psc\text{-}psc} \times (\delta_{GT} / N)$
else $\quad f_{psc\text{-}psc}(\text{A,B}) = f_{s\text{-}psc}(\text{A,B})$

By separating the process of comparing s-psc and psc-psc, we are enforcing the intuition that the comparison between self (e.g. section A) and psc (e.g. psc(B)) should weigh more than that of psc (e.g. psc(A)) and psc (e.g. psc(B)), since here the comparison threshold is raised to $f_{s\text{-}psc}$.

4.2.3 Reference Distribution: f_{rd}

To understand the intuition behind reference distribution, it is important to note that regulations are heavily self-referenced and cross-referenced documents, which contributes to the difficulty in reading and understanding them. Our data, the ADAAG and the UFAS, are heavily self-referenced but not

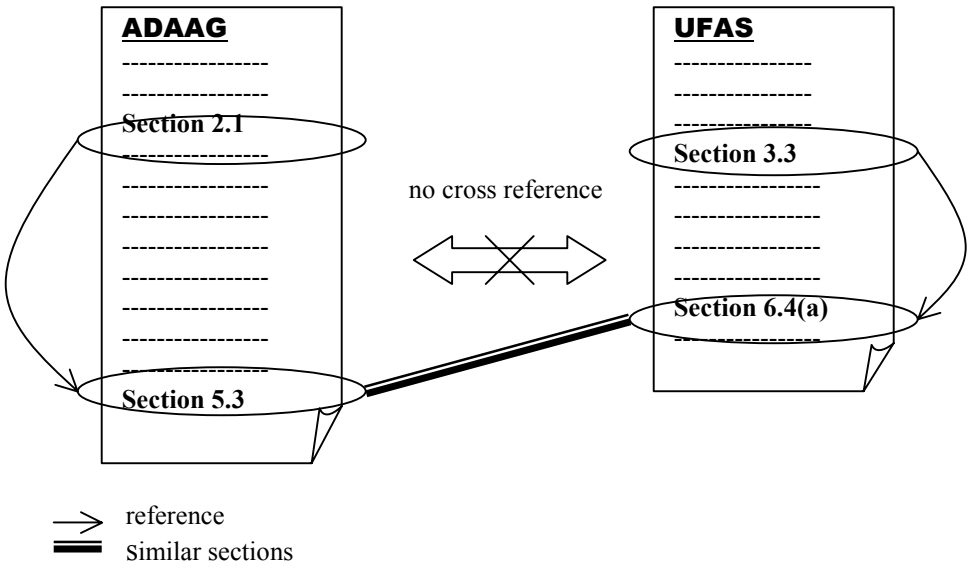
cross-referenced: they reference themselves very frequently, but do not reference outside material as much. For instance, sections in the ADAAG reference other sections in the ADAAG, but do not reference the UFAS nor other documents.

With this understanding in mind, it is easy to explain the process of reference distribution. The hypothesis is that two sections referencing similar sections are more likely to be related and should have their similarity score raised. Therefore, the process of reference distribution utilizes the heavily self-referenced structure of the regulation to further refine the similarity score obtained from section 3.2. The following figure illustrates the idea; it is important to note that we are utilizing the self-reference structure but not cross-reference, which implies that neither the referees nor the referrers are the same for the two sections in interest. One can visualize the problem as separate islands of information: *within* an island information are bridged with references; *across* islands there are no connecting bridges. From Figure 1, it is appropriate to claim that the similarity score between section 2.1 in ADAAG and section 3.3 in UFAS should be increased due to the similarity in the referenced sections. Indeed, this increase should be proportional to the similarity score between the referenced sections.

We deploy a system similar to the s-psc and psc-psc process, replacing psc with refs which represents the set of outlinks from a section:

Set $S = f_{psc\text{-}psc}(\text{refs(A)}, \text{refs(B)})$
N = sizeof(S)
$\delta_{GT} = \Sigma_{s>f psc\text{-}psc(A,B)} (s - f_{psc\text{-}psc}(A,B)), s \in S$
α_{rd} = discount factor of update
if (N != 0) $f_{rd}(A,B) = f_{psc\text{-}psc}(A,B) + \alpha_{rd} \times (\delta_{GT} / N)$
else $f_{rd}(A,B) = f_{psc\text{-}psc}(A,B)$

Figure 1: Comparison of section 2.1 from ADAAG with section 3.3 from UFAS



4.3 Latent Semantic Indexing: f_{lsi}

A traditional LSI approach [3] is used as the control experiment. A term-document matrix [*A*] is populated with the *tf*×*idf* measure of the index term in the document, while documents here represent the entire corpus of sections from both regulations. We have

$$a_{ij} = tf_{ij} \times \log (n / n_i)$$

where tf_{ij} is the term frequency of term i in document (section in our framework) j, and the log term is the inverse document frequency with n being the total number of documents, and n_i being the number of documents with term i. In addition, each document vector is normalized to length one. SVD is then performed on the [*A*] matrix to map index terms to concept space, and also to reduce noise. We have

$$[A] = [P][Q][R]^T$$

after SVD, and the diagonal [Q] matrix is then partly zeroed out for dimension reduction. For some s << r = rank[Q], we take only the largest s singular values from [Q] and zero out the rest to form $[Q_s]$. We then have

$$[A_s] = [P_s][Q_s][R_s]^T$$

with $[P_s]$ and $[R_s]$ being the corresponding stripped out version of the original matrix (since some part of Q is zeroed out). The document-document similarity matrix is given by

$$[A_s]^T[A_s] = [R_s][Q_s]^2[R_s]^T$$

Indeed, we only need the upper right hand quadrant of the doc-doc similarity matrix, since we are only interested in ADAAG against UFAS, but not ADAAG against itself nor UFAS.

4.4 Linguistic Assumptions and Validity

There are a couple points to note here. First, we are only interested in increasing the similarity score between pairs, since this entire comparison stage is a just prequel to conflict identification. Our assumption is, for two sections to conflict with each other, they must be related in the first place, and therefore the process of comparison and ranking. We would like to include as much evidence there is to increase the number of related pairs for conflict analysis, which explains why we are only interested in increasing the similarity score but not decreasing them. For instance, if two sections are entirely the same, but embedded in two completely different neighborhoods, we do not wish to decrease their similarity score.

However, there is a loophole: by increasing the *f* score of all other sections, those who remains the same, i.e. those who does not have as related neighbors or references, are suffering from the same effect of having their *f* score decreased since their ranking will drop inadvertently. Further research is needed to better understand this problem.

5. Implementation[1]

The entire implementation is on Java for its ease of prototyping. The amount of coding is more than expected; it totaled to 1241 lines of code for the entire project. This is mainly due to the unanticipated fact that the presentation style and efficiency are crucial in the system as discussed below. In addition,

[1] Please see also the README file before running the program

a total of 5 different scoring techniques are implemented, with 1 being LSI the control, and the other 4 are all related and sequential; there is no reason to skip any of them.

The first step in the implementation stage is to read in the XML format inputs. Since the XML regulation are more or less well-formed, it is not too difficult if weren't for the fact that the sections are hierarchical. Pointers to its children, as well as to its parent are maintained.

After reading in the input, the system should output information for the user to view and choose from. Due to the sheer volume of the regulation set, it is very difficult to display the result or view a particular section of a regulation. Even when an extremely small set of regulation documents is used the document structure can still be very complicated; e.g. a toy problem like the two regulations in this project (~ 0.5MB) together consist of 900 sections. It is extremely difficult for the user, even for the author, to select sections to view or compare with different techniques. Therefore, certain amount of effort in this project is invested on designing and implementing a Graphic User Interface (GUI) on top of the system. The GUI is minimal since it is not the focus of the project, but serves the purpose for easy retrieval of regulation sections; it also highlights another interesting research issue: how to present a large amount of highly structured text in a user-friendly way. The system is expected to be transported to a web interface in the future.

From the GUI, the user first chooses the input file for the regulations, which leads him/her to the section comparator. The upper half of the panel "Per-section comparison" shows a list of sections in each of the regulations with pull-down menus so the user can choose which section to *view* or *compare* individually. By clicking on *view*, a new window will pop up with the chosen section's information. With *compare*, the chosen pair of sections will be compared and ranked with all 5 scoring techniques. Since the ranking will be given as well with *compare*, it requires all sections to be compared and thus takes a while to run. The α's are parameters that the user can experiment with; all of them should be less than 1 as explained in the algorithm section.

The lower half of the panel "All-section ranking" ranks and returns the top 20 sections according to the scoring technique the user chooses. Again the α's can be adjusted. As discussed in the following "alternatives to improve system" section, the system is modified so that only the first time comparison and ranking takes a while (~2mins), but afterwards selecting other sections to view, compare or other methods to rank should be immediate, unless the α's are changed which forces an update of the computation.

The implementation of the scoring techniques is straightforward except for f_{lsi}. Indexing of terms is done manually by running over the data a first pass to build up the dictionary; then each section is checked against the dictionary to build the word count vector. *Idf* values and vector normalization follow the word count (*tf*) process. The matrix is written out to a file and Matlab is called. After Matlab finishes the computation, it writes the doc-doc similarity matrix to a file, at which point the system reads and outputs the result.

5.1 Filter Usage

Several filters are used in both building the list of index terms, and also in tokenizing the section content. A lowercase character filter, a stopword filter and a Porter's stemming filter are used. The list of 136 stopwords came from CS161 Fall 2000 and is further edited. Before all the filtering work, one set of the regulations has 2940 index terms, which decreased to 1430 afterwards. This amount of

cut is suspicious, and future work should be carried out to investigate whether the system is over-stripping and thus over-generalizing or not.

5.2 Software Usage

5.2.1 Matlab

Matlab is used for the purpose of SVD. It is relatively slow to call Matlab and thus some modification is made; details to follow in Section 5.3. Our data is very sparse; after all the filters (stopword, stemmer, etc.) our term-document matrix is only 1% full. Thus data is inputted as a sparse matrix.

5.2.2 Lucene

The Java package Lucene [5] is used for filtering purposes. Its stopword filter, lowercase filter, Porter's stemmer and tokenizer are used when parsing the text.

5.3 Alternatives tried to Improve the System

Again, just as the unanticipated GUI wrapping of the system, the followings are not the original desired alternative experiments to try to improve the system, as system efficiency is clearly not the focus of this project. However, this provides an insight on dealing with large amount of text documents: running time and space usage are still issues that worth to be thought out.

5.3.1 Slow Execution

The system runs rather slowly; please refer to Section 6.1 for numbers. Quite an amount of change in implementation is experimented to speed up the computation process; for instance, using arrays rather than hashtables as back-end storage; singleton implementation of SectionComparer class so that all comparison results can be re-used in a single instance; no re-computation is performed upon multiple request of comparison or ranking unless the α values are changed; running LSI analysis through Matlab parallel with other processes like f_0 computation; writing out a sparse matrix as input to Matlab instead of a full matrix, and so on. Please refer to the code for details, and to the section of system performance for some timing experiment performed.

5.3.2 Large Memory Usage

Originally several hashtables are used to store the scores f; however, there are over 170,000 elements in each hashtable (pairwise comparison of sections), and the system ran out of memory quickly. The remedy is to re-design the structure so that only one hashtable is needed to store everything, and also a new data structure which encapsulates primitive variables of scores and rankings.

6. System Performance and Results

6.1 Improvements in Running Time

The table below listed some of the modifications in implementation to improve running time. Before modification, it takes around 3 minutes after the first clicking of the "Compare" button to set up f_0, $f_{s\text{-}psc}$, $f_{psc\text{-}psc}$, f_{rd} and f_{lsi} sequentially with hashtable storage. It is painful both for debugging and to use, and also since it takes about the same amount of time to do LSI and the rest of the scoring methods, it is reasonable to try to parallel the operations particularly since Matlab is an external process anyway. Therefore, upon receiving a "Compare" command, the LSI matrix is first set up and Matlab is called. While Matlab is running, all other computation is performed, and the system goes back to check Matlab result after all is done. Quite a significant improvement is shown; about 50% reduction in running time is achieved. However, it still takes about 2 minutes to run "Compare" which is quite slow. Since this project is mostly prototyping, it is anticipated that the computation will be transported to some other software package in the future.

Table 1: Running time comparison of different implementation modifications

	Sequential processing, hashtable used in SectionComparer	Parallel processing, hashtable used in SectionComparer	Parallel processing, array used in SectionComparer
LSI + Matlab (setup time shown for parallel processing)	102 sec	3 sec	3 sec
The rest	102 sec	123 sec	105 sec
Total	204 sec	126 sec	108 sec

6.2 Comparison of Methods

This most difficult part is to come up with a metric to compare and assess the performance of different scoring techniques. Basically certain amount of eyeballing is needed to get a sense of how the system performs.

6.2.1 Per-Section Comparison

From f_0 to $f_{s\text{-}psc}$, some improvement is shown. For instance, section 4.1.6(3)(d) in ADAAG talks about door, while section 4.14.1 in UFAS talks about entrance. Of course, concept match in f_0 could not pick out the similarity between door and entrance, thus $f_0 = 0$. With $f_{s\text{-}psc}$, the system is able to infer some relatedness between the two sections from the neighbors in the tree, and $f_{s\text{-}psc}$ increased to 0.1 with an extreme α value of 1.

From $f_{s\text{-}psc}$ to $f_{psc\text{-}psc}$, very slight improvement is observed. Generally the similarity score does not change much, and it is reasonable. $f_{psc\text{-}psc}$ will not differ from $f_{s\text{-}psc}$ unless the surrounding neighborhoods are highly related while both 1) the pair of sections in comparison is not as related and 2) the pair of sections in comparison is not related to the other half's neighbors, which is unlikely.

From $f_{psc\text{-}psc}$ to f_{rd}, it is disappointing to see that not much improvement is observed. One explanation can be that it is difficult for human to spot out possible sections with helpful outlinks and thus to correctly credit the possible improvement with f_{rd}. Another interpretation is the relatively high threshold in the algorithm: we only update f_{rd} from $f_{psc\text{-}psc}$ if the outlinks have higher similarity between them. First of all, to have related outlinks is already not very likely; and they are compared against the already refined $f_{psc\text{-}psc}$ score, which sets a high threshold for update. Another meaning to this phenomenon is that to have outlinks embedded in sections imply necessarily there are sentences referring to and describing these outlinks. One very common phrase is "… should comply to section x.x.x." From the basic content of the sentences around the outlinks, similarity is captured already by f_0, $f_{s\text{-}psc}$ and $f_{psc\text{-}psc}$, and this relatively high threshold makes reference distribution difficult as our algorithm only updates if the outlinks have higher f scores. This hints a possible refinement in the algorithm used in the future.

Comparing all of the above against LSI, it is clear that neither out-perform the other in all cases. In some cases LSI performs better when the sections have zero references and few or none concepts were tagged by Semio. However, the noun phrases capture sequencing information of words, which is lacking from LSI, which make it the inferior one in some other cases. Another interesting observation is that when two sections share words in common but actually do not mean the same thing, LSI assigns as higher score than the other techniques (e.g. ADAAG 11.2.1(1)(a) vs. UFAS 9.5), which shows that bag-of-word analysis lacks true understanding of the sentence, which hopefully can be captured with

additional concept or domain-specific tags. Clearly we do need bag-of-word information, as well as noun phrases and structures to do a half decent job at comparison.

6.2.2 All-Section Ranking

Clearly all scoring techniques identify different sections as the top 20 most similar pairs of sections. However, if one looks into the sections, the identified pairs are almost all exactly the same; i.e. they have the same exact content. This is not surprised since both ADAAG and UFAS are federal documents, and are expected to be similar.

To further investigate the differences in the 5 scoring techniques on ranking, probably one has to look further down the ranking to check the not exactly the same pairs of sections.

6.3 Ranking vs. raw score

Comparing the different scoring techniques, the first thing to notice is that the similarity scores are surprisingly close. The rankings, on the other hand, are closer when they are ranked higher. When the score approaches zero, the ranking is almost random since there are a lot of ties and the system paid no particular effort to resolve the ties. In addition, the α values are an important factor: scores can be drastically changed by manipulating the α values, but not so much for the ranking.

6.4 Same-Section Ranking Comparison

Another experiment is tried to see how the system performs when the same regulation set is compared against itself; e.g. UFAS vs. UFAS. The system, as expected, ranked the same sections the best; however it is interesting to note that some different sections are actually the same. For instance, according to LSI with $\alpha = 0.1$, sections 4.20.4 and 4.21.4 are the same. Looking into the regulation, this reveals that section 4.20 talks about bathtubs, while 4.21 concerns about shower stalls and the corresponding sections inside are exactly the same.

7. Conclusions and Discussions

It is a shame that the algorithm and implementation works simply took too long and some interesting experiment cannot be carried out due to press of time. For instance, it would be interesting to graph the change of similarity score or ranking against different α values and to experiment with different combination of α's. However re-computation triggered by different α values simply takes too long; further adjustment might be needed in the future. Also a longer list of ranking can be examined, rather than the current pre-set value of the top 20, to give a better comparison of different techniques.

In addition, it would be interesting to try using f_{lsi} as f_0, or a combination of both and thus another α factor to determine the respective weight, to see how the entire refinement process differs with a different starting score.

In conclusion, the developed system is able to read in input files of regulation documents from the user, provide a simple Graphic User Interface to access different sections of regulation, and mechanisms to compare and rank pair of sections according to 5 different scoring techniques, with some α parameters for the user to play around with. From the list of similarity rankings here, we can go on to identify possible conflicting sections by looking further into the sections that are most similar, based on the assumption that conflicting sections are related. A Part-Of-Speech tagger, an interface to WordNet to identify possible conflicting predicate words, and a rule-based system is anticipated in the future for conflict identification.

8. Future Work

Apart from the possible future directions mentioned in the above sections, there are things to be considered more carefully in each of the scoring techniques. In the basis score f_0, two sections can very well have zero concept length, i.e. no concepts are tagged. According to the current implementation, they will have a score of 0; however, another interpretation is possible: since both of them have no concept tagged, this might reflect the fact that they are similar in a lacking sense.

As for the refinement methods, one simple modification is to do iterations. Currently all of the refinement algorithms are only performed once; however there is no reason why one does not iterate till the ranking stabilizes (note that the score f will not converge) other than the fact that time is pressed for the project. Basically, all of the refinement techniques produce a refined score fj based on some initial f_i, and this f_j can be fed back to produce a f_k and so on iteratively.

In addition, reference distribution is initially formalized in a different way. In matrix form, $[F]$ represents the similarity matrix, with rows denoting one regulation while columns denoting another regulation. Thus in our case, $[F]$ is a matrix of size $m \times n$, where m is the number of sections in ADAAG and n is the number of sections in UFAS, and entry F_{ij} is the similarity score between section i in ADAAG and section j in UFAS. Let $[R_1]$ be the matrix of references of the first regulation that is represented as rows in $[F]$, and $[R_2]$ be the matrix of references of the second regulation, which is the columns in $[F]$. Reference matrices are square matrices with rows and columns both representing sections in that regulation, and entry R_{ij} is the number of times section i references section j. In our case, $[R_1]$ is a $m \times m$ reference matrix of ADAAG, and $[R_2]$ is a $n \times n$ reference matrix of UFAS.

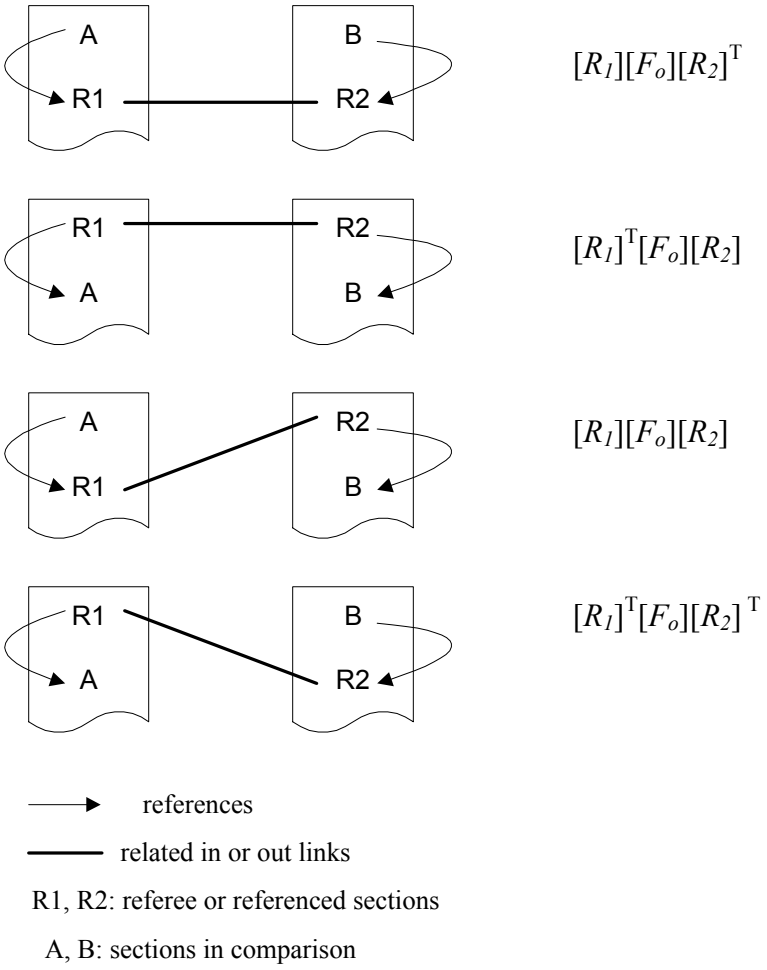
However, with the reference matrix described above, sections with a lot of out references will easily outweigh sections with few out references. Therefore, the reference matrices are normalized so that entry ij represents the number of times section i references section j, divided by the total number of out references from section i.

Our goal is to update the similarity matrix $[F]$ proportional to the similarity score between the referenced sections, and also to the number of times the section is referenced. Denoting $[F_o]$ as the initial similarity matrix obtained from, e.g. section 4.1 or 4.2.1, $[F_1]$ as the updated or refined similarity matrix after reference distribution, and α as the proportion of the update from references, we have

$$[F_1] = [F_o] + \alpha [R_1][F_o][R_2]^T$$

From here it is easy to see that terms like $[R_1]^T[F_o][R_2]$, $[R_1][F_o][R_2]$ and $[R_1]^T[F_o][R_2]^T$ make sense as well as $[R_1][F_o][R_2]^T$. The assumption that sections referencing related sections are similar can be expanded to say that sections being referenced by related sections are similar, and for other combinations of reference-ers and referees. This is best illustrated with a picture; please see Figure 2 below.

Figure 2: Comparison of sections A and B using f(R1, R2)



There is one subtlety in this formulation; namely sections having zero outlink are at a disadvantage here. Normalization of the reference matrix ensures that a section having 5 outlinks will not be outweighed by one with 7; however there is no normalization one can perform on 0 out-referencing sections. This issue remains to be solved.

9. References

1. The Access Board. http://www.access-board.gov/
2. Baeza-Yates, R., Ribeiro-Neto, B. Modern Information Retrieval. *Addison Wesley Ltd.*, 1999.
3. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., and Harshman R. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 1990. http://citeseer.nj.nec.com/deerwester90indexing.html
4. Gibbens, M. California Disabled Accessibility Guidebook 2000 (CalDAG 2000). *Builder's Book, Inc.*, 2000.
5. Jakarta Lucene. http://jakarta.apache.org/lucene/