DISSERTATION PROPOSAL

REGBASE: An Information Infrastructure for Regulation Management and Comparative Analysis

Gloria T. Lau

Stanford University

Sept 10th, 2002

Contents

1. Introduction

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, to locate a particular regulation of interest is no easy matter: neither a generic bookstore nor a public library has a good chance of possessing such a regulation. In addition, even with the hardcopy of the regulation in hand, an average citizen is likely to soon get lost in the jungle of definitions of legal terms and references.

Apart from the general use by the public, regulations are mostly reviewed and used by industry designers, planners and inspectors. Industrial productivity can be greatly increased if tools are provided to aid understanding of regulations. For instance, the permitting process in the design and construction industry is significantly prolonged by the increasing complexity of regulations. Designers, albeit more knowledgeable than the general public, have yet to search through the continuously changing provisions and locate the relevant sections related to the project, then sort through potential ambiguities in the provisions. Inspectors have to go through a similar evaluation process before a permit can be approved.

Besides the difficulties in locating and understanding a particular regulation, the inherent nature of multiple coding agencies also deserves attention. Regulations are typically specified by Federal as well as State governmental bodies and are amended and regulated by local counties or cities. These multiple sources of regulations sometimes compliment and modify each other, and users often have to choose the more restrictive code provision as the safest route. However, there are instances where the provisions of two applicable codes are in direct conflict. Designers often turn for resolution to reference handbooks that are independent of governing bodies, such as the California Disabled Accessibility Guidebook (CalDAG) [7] by Gibbens. The regulations, amending provisions and interpretive manuals together create a sheer volume of semi-structured documents with possible differences in formatting, terminology and context.

The advance in technology has provided us with such tools to mitigate some of the above problems. It is now easier to locate and search through a particular regulation; although some of them are still only available as hardcopies, most of them are slowly migrating to digital format, e.g. the 2000 International Building Code (IBC) [8] is available on CD-ROMs with the provisions in Portable Document Format (PDF). Some are available online in Hypertext Markup Language (HTML). With the recent development of the Extensible Markup Language (XML), one can consolidate different regulations and represent codes in a semi-structured format. To enhance understanding of regulations, textual comparison techniques from the field of Information Retrieval (IR) can be deployed. It is anticipated that information technology (IT) can help solve some of the problems in locating, reasoning through and comparing different sets of regulations.

1.1 Motivation

This section presents several motivating examples from the Americans with Disabilities Act (ADA) Accessibility Guidelines [1] and the California Building Code (CBC). Some of the examples come from Gibben's CalDAG [7]. Together they explain why a framework is needed to locate similar or related code sections, and to compare and contrast them if possible.

The following examples from the ADA Accessibility Guidelines (ADAAG) [1] illustrate some of the research issues including the complexity and ambiguity of provisions. For the first example, the difficulty in understanding lies in the cross references and multiple requirements depending on the number of toilet stalls provided. It shows that individual provision is not self-contained and the information is incomplete due to the existence of out references. The latter section in Example 1 presents a vague provision with the term *easy to grasp* subject to interpretation, as well as an asterisk denoting yet more advisory material in the Appendix of the ADA.

Example 1

> 4.22.4 Water Closets. If toilet stalls are provided, then at least one shall be a standard toilet stall complying with 4.17; where 6 or more stalls are provided, in addition to the stall complying with 4.17.3, at least one stall 36 in (915 mm) wide with an outward swinging, self-closing door and parallel grab bars complying with Fig. 30(d) and 4.26 shall be provided. Water closets in such stalls shall comply with 4.16. If water closets are not in stalls, then at least one shall comply with 4.16.
>
> 4.13.9* Door Hardware. Handles, pulls, latches, locks, and other operating devices on accessible doors shall have a shape that is *easy to grasp* with one hand…

Numerous instances exist where one code is more restrictive than another; here we take the regulation over curb ramps in accessible parking stalls as an example below [7]. The CBC allows curb ramps encroaching into accessible parking stall access aisles, while the ADA disallows encroachment into any portion of the stall. The example from the CBC also demonstrates another research issue related to regulations: general rules stated in the provision are modified by exceptions or further specifications, which makes understanding the code a non-trivial task.

Example 2

> ADAAG Appendix
> A4.6.3 Parking Spaces. …The parking access aisle must either blend with the accessible route or have a curb ramp complying with 4.7. Such a curb ramp opening must be located within the access aisle boundaries, not within the parking space boundaries. Unfortunately, many facilities are designed with a ramp that is blocked when any vehicle parks in the accessible space. Also, the required dimensions of the access aisle cannot be restricted by planters, curbs or wheel stops.
>
> CBC

1129B.4.3 …Pedestrian ways which are accessible to persons with disabilities shall be provided from each such parking space to related facilities, including curb cuts or ramps as needed. Ramps shall not encroach into any parking space.
EXCEPTIONS: 1. Ramps located at the front of accessible parking spaces may encroach into the length of such spaces when such encroachment does not limit the capability of a person with a disability to leave or enter a vehicle, thus providing equivalent facilitation. See Figures…

An example of directly conflicting provisions is presented below, with the ADA focuses on wheelchair traversal while the CBC focuses on the visually impaired when using a cane. Gibbens [7] points out that "when a state or local agency requires you to construct the California required ½" beveled lip, they are requiring you to break the federal law", and this clearly should be brought to the user's attention.

Example 3

ADAAG
4.7.2 Slope. Slopes of curb ramps shall comply with 4.8.2. The slope shall be measured as shown in Fig. 11. Transitions from ramps to walks, gutters, or streets shall be *flush* and *free of abrupt changes*. Maximum slopes of adjoining gutters, road surface immediately adjacent to the curb ramp, or accessible route shall not exceed 1:20.

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.

1.2 Objectives

The goal of this thesis is to develop an integrated framework for regulation representation, analysis and comparison. In order to develop a prototypic system, we focus on accessibility regulations, whose intent is to provide the same or equivalent access to a building and its facilities for disabled persons. Our corpus currently includes the ADAAG and the Uniform Federal Accessibility Standards (UFAS) [2]; both of which are Federal documents that provide prescriptive measures on disabled access. We anticipate the addition of Chapter 11, Accessibility, of the IBC [8] to our corpus in the near future. The handbook by Gibbens is also a good document for comparison; however it is only available in hardcopy format and we have yet to convert it to digital form. Once the prototype is thoroughly tested on accessibility regulations, we would like to incorporate environmental regulations as well to demonstrate scalability and practicality of our system.

This study aims to produce a system where one can easily assess different regulations in a consolidated format, trace out references within provisions, search and view relevant sections. A taxonomy will be developed on top of the regulations, and users can browse through regulations categorized according to the hierarchy. Upon the selection of a section by the user, comparison of other provisions will be performed and related sections are returned to aid understanding. Potential contradicting provisions are analyzed and brought to the user's attention as well.

1.3 Scope and Approach

The organization of this thesis proposal is as follows. First, a brief literature review is presented in Chapter 2. Section 2.1 discusses previous work relating to the development of the repository. This chapter also deals with the particularity of regulatory documents in Section 2.2, which explains why traditional text mining algorithms cannot be the sole solution to our problem.

The first phase of the project, which is discussed in Chapter 3, is to develop an online repository of regulation documents. Difficulties in properly populating the repository include the initial non-digital format of some government regulations, and different original formats, e.g. PDF or HTML, in the case of already-digitized documents. Data format conversion is performed by a shallow parser which results in properly structured regulations in XML. The data are then passed to Semio Tagger[1] [18], a software for noun phrase extraction and document categorization. To ease browsing, an ontology is developed on top of the concepts extracted by Semio, where users can view documents categorized according to the ontology.

Also included in Chapter 3 is the process of feature extraction. Features are the key element to regulation analysis in Chapters 4 and 5, and they come from both the regulatory corpus and reference materials such as engineering handbooks. All of the extracted features are integrated as additional tags to the documents, and the resulting refined regulations are the point of departure for the analysis core.

With a fully functional online repository and additional features extracted, analysis of regulations is performed. The similarity analysis core is presented in Chapter 4, which defines the similarity score between pairs of provisions and successive refinements of the score. Briefly speaking, the base score is computed by matching features, and is further refined by taking into account the scores of neighboring provisions. Reference distribution completes the process by combining scores of linked sections. In addition, the performance of our system is compared with that obtained using Latent Semantic Indexing, LSI, technique.

Upon the completion of similarity evaluation, conflict analysis is performed and the details are discussed in Chapter 5. The conflict analysis core takes only the most related pairs of provisions to check for conflicts, and a sequence of tools, e.g. part-of-speech tagger, WordNet, and etc., are used to aid both automated conflict rule generation and human-written rules. A list of possible conflicting sections is returned as a result.

It is worth noting that in the course of the development of the system, we aim to be as general and scalable as possible so that our framework can be tested outside the very narrow field of accessibility regulations. Intention to work with environmental and other regulations is clear, and focus on system scalability is anticipated in the future. However, a minimal amount of specific techniques, which are only applicable on this regulatory domain, are unavoidable.

[1] Semio Tagger is a trademark product by Semio Corporation.

2. Related Work

Guidance to interpretation of legal documents has existed as long as legal documents themselves. Reference materials and handbooks are merely the byproducts of the many sources of regulatory agencies and the ambiguity of regulatory documents. For instance, CalDAG is a handbook written for compliance guidance with the accessibility code. It claims to "sort out and explain the differences between the ADA & Title 24 that all California professionals must understand and apply to comply with both laws" [7].

Despite the fact that interpretive guidelines have long existed, the introduction of information technology to aid legal interpretation is rather new. The recent increase in network capacity has given rise to the proposal of a web-based broker for regulations [10]. Data mining techniques, in particularly text mining algorithms are sought to perform classification and clustering on legal documents [19]. Most of the recent research focus on enhancing the *search* and *browse* aspect of legal corpus, whose targeted users are legal practitioners.

To aid legal *reasoning* and *interpretation*, most knowledge bases develop upon a rule based system or a network representation. However, rule based system is always criticized for its lack of flexibility, especially in logic programming, to accommodate the frequent ambiguity and vagueness in legal issues. Graph or network representation, on the other hand, requires knowledge engineers and domain experts to create the representation structure themselves, which is often a difficult and subjective task [19]. Therefore, during the development of our system, assumptions involving the ambiguity of law or the structure of the model are avoided.

2.1 Building the Repository

Feature extraction is an important step in repository development when the data dimension is large. It is a form of pre-processing, e.g. combining input variables to form a new variable, and most of the time features are constructed by hand based on some understanding of the particular problem being tackled [4]. Automation of this process is also possible; in particular, in the field of information retrieval, software tools exist to fulfill "the task of feature extraction … to recognize and classify significant vocabulary items" [4]. The IBM Intelligent Miner for Text [6] and the Semio Tagger [18] are both examples of fully automated key phrase extraction tools.

Apart from reducing the effect of the curse of dimensionality, feature extraction in text mining identifies important phrases by pulling together terms to form *concepts*, in Semio language. This captures the sequencing information of terms, and experiments have shown that phrases can convey more important information than the terms separated. For example, "*joint venture* is an important term in the Wall Street Journal database, while neither *joint* nor *venture* are important by themselves. In fact, in a 800+ Mbytes database, both *joint* and *venture* would often be dropped from the list of terms by the system because their idf weights were too low" [12].

In addition to data cleaning and pre-processing, the repository should be equipped with a search and browse capability for viewing and retrieval of documents. Due to the recent bloom of the web, an extensive amount of research focusing on retrieving more relevant documents based on keyword search has been done. Well-established techniques such as query expansions have been deployed to increase retrieval accuracy. Therefore in our project, we assume that at least one relevant document will be located by the user either with keyword search or by browsing through the Semio Taxonomy interface. From there, related documents are suggested to the user by our system; in a sense we focus on refining the back end comparison technique for documents rather than matching queries at the front end.

2.2 Analyzing the Documents

Before surveying on traditional textual similarity analysis techniques, it is worth noting that legal documents are indeed very different from typical documents found in generic text corpora. Regulatory documents possess three main characteristics that are not found in general text:

(1) They are well structured and follow a strict hierarchy of parent and child provisions.
(2) Sections are heavily cross-referenced.
(3) Important terms are defined in a relatively early "definition" chapter of the regulation.

This diversion from generic documents makes regulatory materials interesting to analyze, and is the main reason why traditional textual comparison techniques need to be refined to apply on regulatory domain. Combination of different techniques is sought to further utilize the above characteristics of legal documents to improve analysis result. Below a brief overview of related textual and structural analysis algorithms is given.

Text document comparison, in particular similarity analysis between documents is widely studied in the field of information retrieval. Techniques such as the Boolean model and the Vector model exist [3], and most of them are bag-of-word type of analysis. This type of models cannot capture synonymic information without the help of thesauri, and Latent Semantic Indexing (LSI) [5] 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 synonyms that represent the same concept are mapped onto the same concept axis. In our project, LSI will be used as the control experiment to compare with.

The heterogeneity of different data structures and their implied comparisons have been widely studied in the field of database management systems. In particularly, semantic interoperations between sources of information are enabled by a well-defined ontology mapping system [16]. However, as pointed out in the above three characteristics of regulatory documents, all regulations follow a strict hierarchical structure regardless of their source. In addition, the terminologies used in each regulation are well defined, which makes the use of an ontology matching system unnecessary. In the future, if more free-formed texts are added to the corpus, or if the relationships between provisions

become more complicated than parents and children, an ontology matching system can be handy.

Graph theory has been widely studied and algorithms are developed to match vertices according to different criteria. If we cast section as node, and regulation as a collection of nodes, a bipartite graph can be formed by linking the two disjoint sets of nodes with pairwise section similarity. In particular, the stable marriage algorithm finds the optimal match between nodes. However, finding the most likely match for each section does not solve our problem; two sections identified as matching should not prevent them to be matched with other sections.

In addition to comparing the body text of provisions, the heavily referenced nature of regulations provide extra information about provisions, and link analysis comes in handy to refine the 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, while our problem here are separate *islands* of information where *within* island documents are highly referenced; *across* island they aren't. Thus a different algorithm is sought to better serve the need.

Finally, definitive and glossary terms in regulations associate a term with its definition in the specific regulatory domain. This resembles dictionary definitions used in word sense disambiguation tasks, where one counts the texts in sense definition as extra evidence that the word appears as a particular sense [14]. Another approach to word sense disambiguation is to compute the distance between context vectors [11]; a context vector is the frequency of words appearing in the context window of the word to be disambiguated. Therefore by computing the distance between context vectors, we gain a sense of the closeness between two words. Nevertheless, this approach cannot be applied directly on glossary terms since our goal is to obtain similarity between sections but not definition terms.

3. Building the Repository

3.1 Introduction

The first phase of the project is to develop an online repository of regulation documents, and a schematic of the process is shown in Figure 1. Regulations originally come in different formats, e.g. HTML, PDF, plain text, and so on; thus a shallow parser is needed to consolidate different formats to our selected standard, XML. Brief justification of why XML is selected is given in Section 3.2, while the shallow parser is described in Section 3.3.1. After the documents are parsed into XML format, a software tool called Semio Tagger is used to help extraction of key concepts as well as the development of a taxonomy for viewing purpose; please see Section 3.3.2 for details.

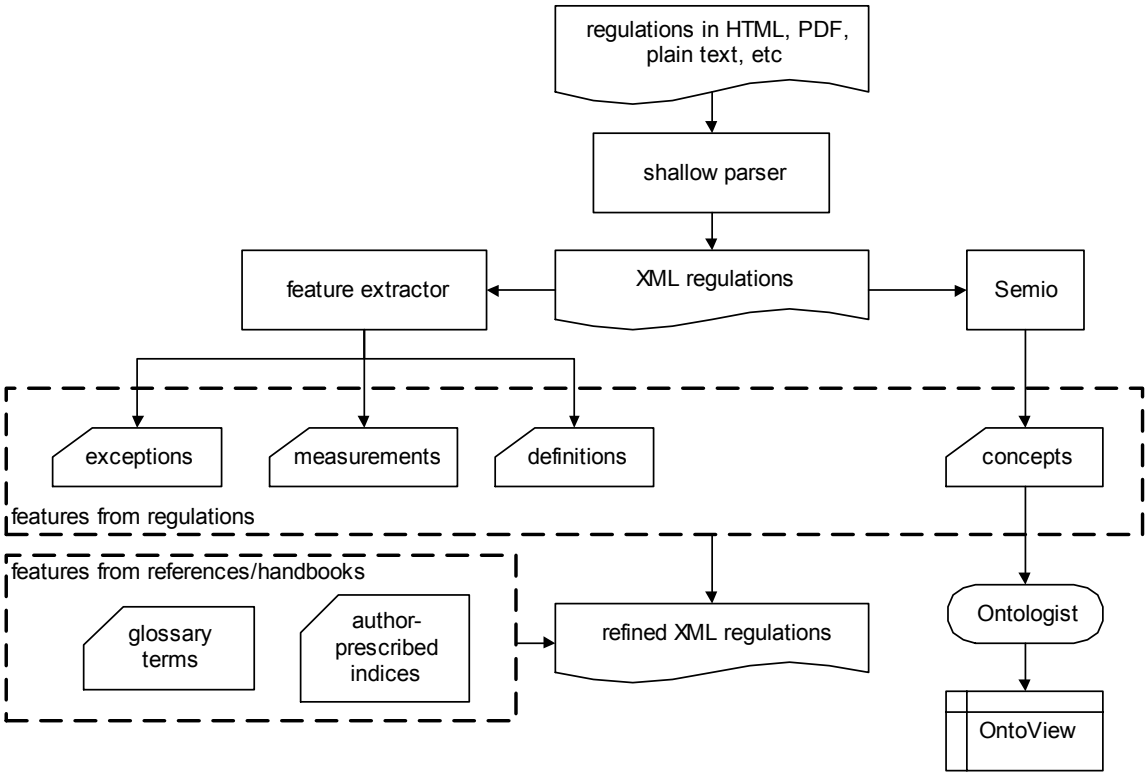


Figure 1: Repository Development Schematic

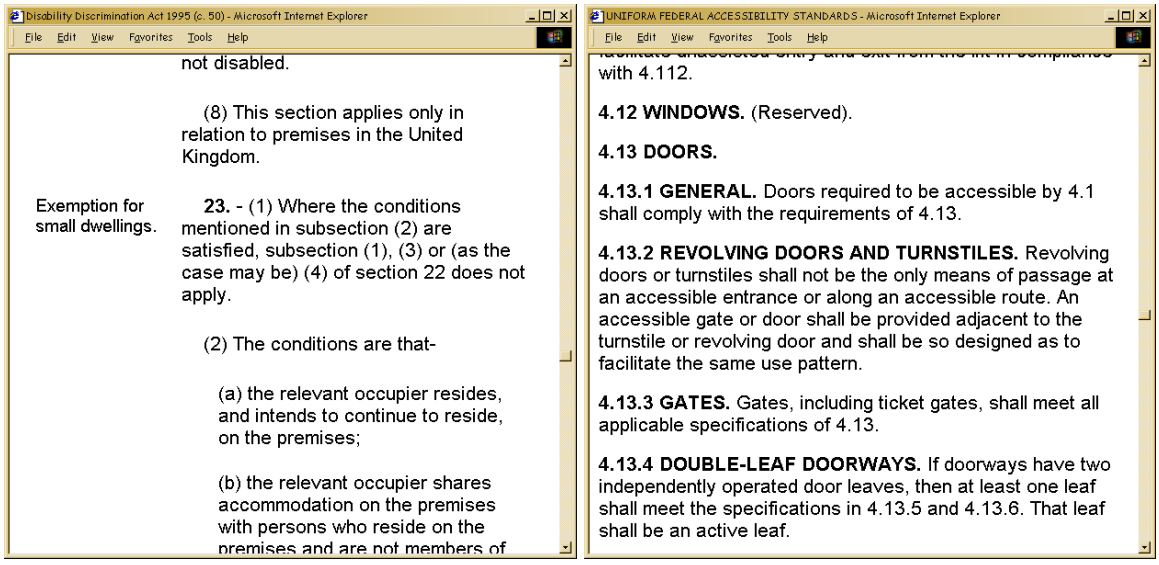
Section 3.4 deals with feature extraction, which is an important process to aid future document analysis. Two different sources of features, namely features from within the regulation corpus and features from outside, are extracted with the help of software tools and parsers developed for this task. All of the features are added to the documents as extra tags, and the refined XML regulations are produced as a result. Both the similarity analysis core, which is discussed in Chapter 4, and the conflict analysis core, which follows in Chapter 5, expect the refined regulations as an input.

Most of the proposed repository development in this Chapter is completed. The shallow parser has been implemented in Perl, and is currently being reformatted and transported to Java by Ms. Pooja Trivedi. The taxonomy is to be contructed using Semio with the help from Mr. Charles Heenan. As for feature extraction, all within-regulation features showed in the upper box in Figure 1 are already implemented, while the features from outside domains showed in the lower box have yet to be added to the repository.

3.2 Overview of the Current Standard

A brief survey of the current standard for regulation documents shows that a consolidated format is needed. Some of the regulations, e.g. the ADAAG and the UFAS, are provided in HTML format which is unstructured and bundles data with style. Some are stored as either a single-columned or two-columned PDF, for instance the IBC [8]. Others are available as hardcopy only such as the CalDAG [7].

In addition, just as every writer has one's own style, the format differs even within a group of pure HTML or PDF regulations. Figure 2 shows an example of two regulations in HTML. The first major difference between the UFAS and the UK Disability Discrimination Act (DDA) is the section numbering scheme where one uses full path like 4.2.1 while the other lists only the current location, e.g. listing only the number 2 instead of 4.2.1. They also differ in styling: the entire UK DDA is written as a HTML table with the first column being the section title if there is any, and the second column is the main text. The ADAAG follows a traditional HTML style.



(a) UK DDA (b) ADAAG

Figure 2: Differences between regulations in HTML

The key issue here is to extend the usability of digital information: a standard format for interoperable information exchange is needed to consolidate regulations from different places. XML is chosen as the communication model because of its expressiveness to represent the organization of provisions, its ability to format semi-structured data and its flexibility in tagging compared to HTML. For instance, the tree hierarchy of regulations can be captured by properly structuring XML tags, and also tables in HTML can be directly embedded in XML.

3.3 Basic Browse and Search

3.3.1 The Shallow Parser

Data cleaning and consolidation can easily account for more than 80% of the total data mining time especially when there exist multiple data sources; therefore, to minimize effort, a shallow parser is developed to transform regulations in HTML to XML. It is expected that the parser will be generalized to transform PDF into XML as well. Currently it takes an HTML regulation as input (e.g. ADAAG or UFAS), and outputs the corresponding well-formed XML file, with added tags and removed HTML formatting. Please refer to the Appendix for further details in implementation and tag selection.

The basic XML structure is as follows: each section is a `regElement` tag, with parents and children to represent the hierarchy. For instance, section 13 is the parent of section 13.1 and 13.2, and 13.2 in turn is the parent of 13.2.1 and 13.2.2. Essential information such as the section number, the section title are extracted as well; an example `regElement` is shown below. Provisions with an asterisk denote that related material is present in the Appendix of the regulation, which cannot be overlooked in comparative analysis.

```
<regElement name="ufas.4.32.1" title="minimum number" asterisk="0">
   <regText> Fixed or built-in seating, ... </regText>
   <ref name="ufas.4.1" num="1" />
   <ref name="ufas.4.32" num="1" />
</regElement>
```

The body text of each provision is captured in the `regText` tag which is an element of the `regElement` tag. To aid user retrieval of related regulations and for future analysis purpose, out references from the provision and the number of references are also extracted as the `ref` tag. Note that the above is neither a complete list of the tags nor the complete functionality of the parser, since more crucial elements are added in the feature extraction phase.

The resulting XML regulation is rendered with a stylesheet in Extensible Stylesheet Language (XSL). Sections are separated by a horizontal bar, and references can be followed by clicking on the link in the body text. Please see Figure 3 for a sample output.

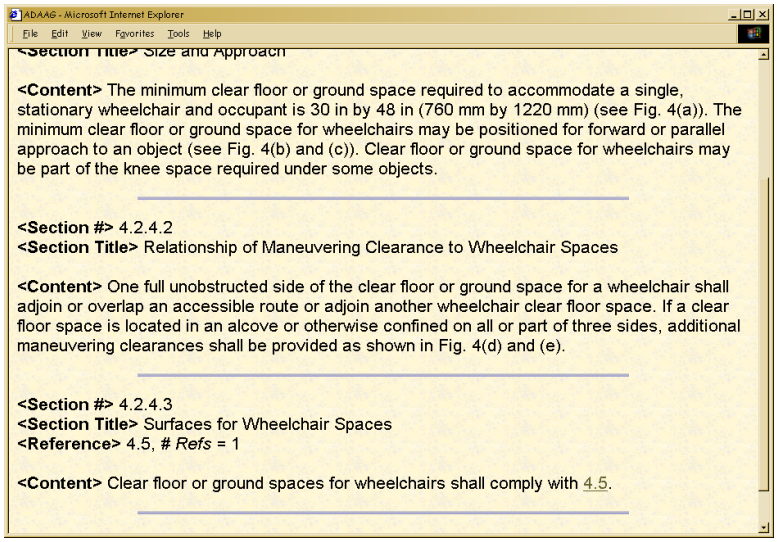


Figure 3: Output from the shallow parser rendered in XSL

3.3.2 Browse and Search with a Taxonomy

Other than browsing through the regulations by provisions as discussed in the above section, it is helpful to provide a taxonomy for users to view the documents categorized according to the hierarchy. Semio Tagger is one software among many others, that provide such capability. It first identifies a list of noun phrases, or concepts in Semio terms, that are central to the corpus. We feed Semio Tagger with our repository of regulations, and get in return a list of concepts; a concept latching tool is provided by Semio to help knowledge engineer to categorize the concepts and create a taxonomy. Documents are thus categorized according to the taxonomy, and users can click through the structure to view relevant provisions classified with concepts. Figure 4 below shows a sample taxonomy generated using Semio.

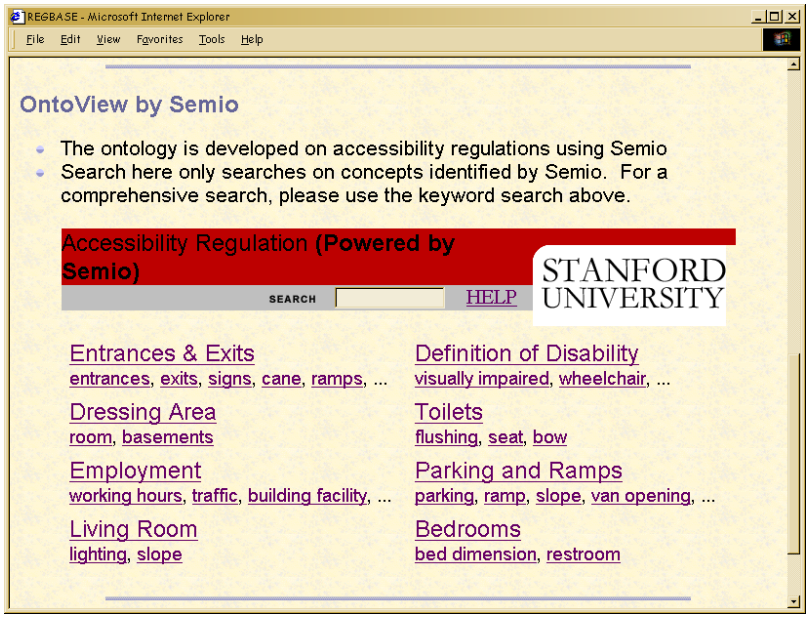


Figure 4: OntoView by Semio

From Figure 4, we can see that Semio also provides search capability; however this is a concept search that only searches on the concepts extracted by Semio. Therefore, to allow for a complete keyword search, we plan to use a public domain search engine mounted on the corpus.

3.4 Feature Extraction for Comparative Analysis

After the data cleaning, consolidation and categorization processes, the repository is complete for viewing and provision retrieval. However, several features need to be added to the repository before we can start to analyze and compare provisions. These extra evidences for comparison are represented as additional tags in provisions.

This process extracts from regulations the identified features that signal related or similar sections. Some of the features can be applied generically on other sets of regulations, while some are specific to the domain of accessibility; for instance, numeric `measurement` might only make sense in the domain of disabled access code but not in human rights law. In addition, what defines *evidence* in a certain domain of regulations is also subjected to the knowledge engineer's judgment. In this context, we strive to be as generic as possible, and all of the extracted features can be easily extended to other engineering domains as well.

3.4.1 Concept Tag

Traditional Boolean model or Vector model in the field of information retrieval (IR) provides a mechanism for text analysis. Indexing the texts using all of the words, except stopwords, generates a huge multi-dimensional space with one axis representing one

word. Using singular value decomposition, in short SVD, as the dimensional reduction tool, similar words are pulled together as one reduced axis. However, it is still computationally intensive to perform SVD, and the initial sparseness of the matrix is destroyed after dimension reduction. In order to seek an alternative to the bag-of-word vector model and the SVD technique, *concepts*, or key phrases, are used due to its simplicity compared to the traditional index terms and its ability to capture sequencing information on words.

As mentioned above, Semio Tagger is run against our corpus and a list of concepts that Semio identifies as important is returned. In our case, the ADAAG and the UFAS together generate just over a thousand concepts. The parser takes this list of concepts as an input and tags each provision with a `concept` tag and the corresponding count of appearances of the concept (`num`) as shown below. To increase the number of matches, both the concepts and the texts in the provision are stemmed with Porter's Algorithm [17] before matching. Below is an example of a concept and its count.

```
<concept name="stationary wheelchair" num="2" />
```

3.4.2 Author-Prescribed Indices

Semio extracts key phrases from the corpus by linguistic analysis and other techniques; these machine-generated phrases are a good measure of important concepts in the body text of provisions. Another source of potentially important phrases comes from author-prescribed indices at the back of reference books or even the regulation itself; this type of human-written information sometimes can be more valuable than machine-generated phrases.

To start out, index terms from Chapter 11, Accessibility, of the IBC [8] will be tagged with the repository. The process is identical to that of concept tagging: a list of index terms is submitted to the parser, which checks for matches with the body text in the provision. Again the terms and the body texts are both stemmed to increase the number of matches. Below is an example of the `indexTerm` tag.

```
<indexTerm name="valet parking" num="1" />
```

3.4.3 Definition and Glossary Tags

In regulation documents, often there is a designated section in an early chapter that defines the important terminologies used in the code, such as Section 3.5 in the ADAAG. These human-generated terms are more likely to convey key concepts than machine extracted ones such as Semio concepts. In addition, the definition of a term gives the *meaning* to a term, which is useful in comparison.

```
<definition>
   <term> Accessible </term>
   <definedAs> Describes a site, building, facility, or portion thereof
   that complies with these guidelines. </definedAs>
</definition>
```

Similarly, engineering handbooks always define the important terms used in the field in the glossary. For instance, the Kidder-Parker Architects' and Builders' Handbook provides an 80 pages glossary that defines "technical terms, ancient and modern, used by architects, builders, and draughtsmen" [13]. The difference between `definition` and `glossaryDef` is that `definition` comes from the regulation itself, while `glossaryDef` comes from sources other than the regulation.

```
<glossaryDef>
   <term> Return Head </term>
   <definedAs> The continuation of a molding, projection, etc., in an
   opposite direction. </definedAs>
</glossaryDef>
```

3.4.4 Exception Tag

As shown in the example of Section 1.1, an exception amends the body text of a provision. It can be regarded as part of the body text; however, this does not help comparison later on since exceptions are negated provisions and should not be mixed with the body text. Therefore, they are captured in an `exception` tag as follows.

```
<exception>
   Ramps located at the front of accessible parking spaces may encroach
   into the length of ...
</exception>
```

3.4.5 Measurement Tag

In accessibility provisions, measurements play a very important role; in particular they define most of the conflicts, e.g. one provision might ask for a clear width of 10 to 12 inches, while another one might require 13 to 14 inches. Therefore, it is crucial to identify measurements and the associated quantifiers if there is any. In our context, measurement is defined to be length, height, angle, etc. They are numbers preceding units. Quantifiers are noun phrases that modify a measurement, e.g. at most, less than, maximum, etc. They can be reduced to a root of either *max* or *min*, e.g. at most and less than are maximum requirements, thus reduce to *max*.

Similar to concept tagging, our parser takes a list of units, quantifiers and their roots as input. This list can easily be generated by a knowledge engineer or a careful reader of the regulation. From the ADAAG and the UFAS, we have the following:

(i) Units: inch, foot, degree, second and pound.
(ii) Quantifiers: minimum, maximum, at least, at most, higher than, greater than, more than, less than, steeper than, fewer than, faster than, or less.

The parser first identifies numbers followed by units, e.g. the number 2 followed by the unit lbf as in 2 lbf, and tags the regulation with this additional `measurement` tag. Quantifier is an optional attribute in the `measurement` tag and is identified if it appears in

the vicinity of the measurement; the trial window width is set to be 5 words in front and after the measurement within the same sentence. Negation, if appearing right in front of the quantifier, is extracted as well and the final quantifier is reduced to its root *max* or *min*. An example is shown below.

```
<measurement unit="lbf" size="2" quantifier="max" />
```

In addition, range (e.g. 2 to 3 inches) and area (e.g. between 2 and 3 lbf) measurements are identified, and an area measurement tag is shown as follows:

```
<measurement unit="lbf" size1="2" size2="3" area="1" quantifier="min"
/>
```

3.4.6 Examples with Complete Mark-up

Presented below are two examples with the complete set of mark-ups. The first example comes from the ADAAG, and it shows the definition mark-up together with some `concept` and `indexTerm` tags extracted. The second example is a typical provision from the UFAS, which contains `exception`, `measurement` and `ref` tags in addition to the body text `regText` tag. To illustrate most of the mark-ups in a single provision, the selected provisions tend to be rather lengthy, and therefore only excerpts of the body text are shown below.

Example 1

Original section 3.5 from the ADAAG

3.5 DEFINITIONS.

…

ACCESSIBLE.

Describes a site, building, facility, or portion thereof that complies with these guidelines.

…

CLEAR.

Unobstructed.

…

Refined section 3.5 in XML format

```
<regElement name="adaag.3.5" title="definitions" asterisk="0">
   <concept name="accessible means" num="2" />
   <indexTerm name="facility" num="1" />
   <definition>
      <term> accessible </term>
      <definedAs> Describes a site, building, facility, or portion
      thereof that complies with these guidelines. </definedAs>
   </definition>
   <definition>
      <term> clear </term>
      <definedAs> Unobstructed. </definedAs>
   </definition>
</regElement>
```

Example 2

Original section 4.6.3 from the UFAS

4.6.3* PARKING SPACES. Parking spaces for disabled people shall be at least 96 in (2440 mm) wide and shall have an adjacent access aisle 60 in (1525 mm) wide minimum (see Fig. 9). Parking access aisles shall be part of ...

EXCEPTION: If accessible parking spaces for vans designed for handicapped persons are provided, each should have an adjacent access aisle ...

Refined section 4.6.3 in XML format

```
<regElement name="ufas.4.6.3" title="parking spaces" asterisk="1">
   <concept name="access aisle" num="3" />
   <indexTerm name="accessible circulation route" num="1" />
   <measurement unit="inch" size="96" quantifier="min" />
   <ref name="ufas.4.5" num="1" />
   <regText> Parking spaces for disabled people shall ... </regText>
   <exception> If accessible parking spaces for ... </exception>
</regElement>
```

3.5 Summary

In this chapter, we walk through the process of data format consolidation, data browsing and searching, and feature extraction. A shallow parser is developed to transform regulations in HTML format to XML format; in the future regulations or complimentary documents in PDF format should be consolidated to XML format as well. To ease user browsing and searching over the corpus, a taxonomy is developed by a knowledge engineer with the help of a software tool called Semio Tagger. Important concepts are extracted by Semio, and the knowledge engineer takes this list of concepts to build a classification hierarchy of relevant documents by using a concept latching tool from Semio. The user can then search by keyword or concepts for interested documents, or go through the taxonomy and find related documents classified according to the hierarchy.

Once the data is consolidated, the repository is refined with the addition of features, which define an important starting point for both similarity and conflict analysis cores. Features from within the regulation corpus as well as features from outside references or handbooks are both extracted and added to provisions as extra mark-ups. The resulting refined XML regulations, as shown in the above mark-up examples, are used as inputs to both analysis cores.

4. Similarity Analysis

4.1 Introduction

With the repository fully developed, users can browse and search through regulations rather easily. However, upon finding a relevant provision for a particular design scenario, clients have to search multiple codes with multiple terms to locate yet more related provisions if there is any. Thus, our goal is to provide a reliable measure of relatedness of pairs of provisions, and to suggest similar sections of a selected provision based on the similarity measure. Figure 5 shows a modified version of Figure 3 where one can browse through each provision and its suggested related sections by tracing out the links. The recommended sections are ranked according to the relatedness to the current provision.

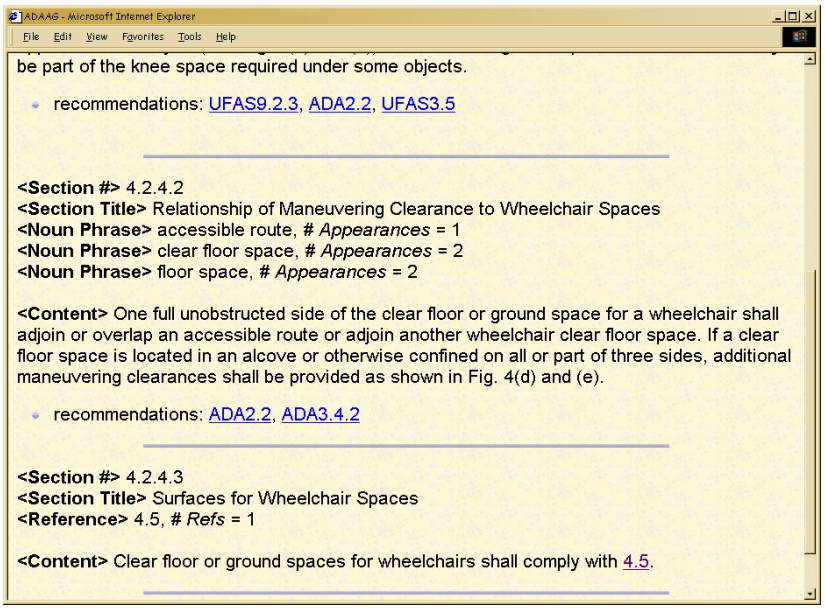


Figure 5: Related sections linked to each provision

Here since a typical regulation can easily go over thousands of pages, we do not attempt to compare a full set of regulation against one another; rather a *section* from one set of regulation is compared with another *section* from another set, such as a comparison between section 4.3(a) in ADAAG and section 3.12 in UFAS. The comparison core, which is shown on the left in Figure 6, can also be used to compare sections from the same regulation, but for now only different regulations are compared.

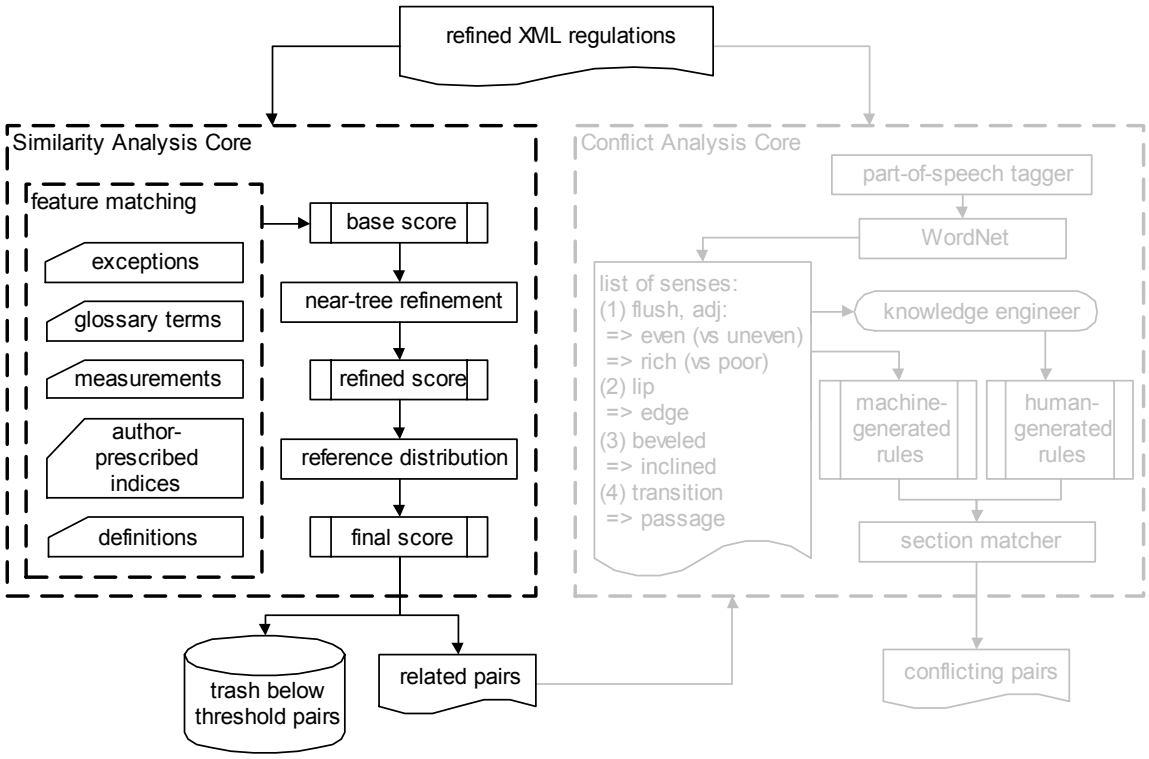


Figure 6: The Analysis Core Schematic

The similarity analysis core takes as an input the refined regulations and the associated features from the repository development engine. It produces as a result a list of the most similar or related pairs of provisions, which is expected by the conflict analysis core showed dimly on the right in Figure 6. The similarity core first computes a base score for each pair of provisions by matching extracted features. It then refines the base score by including the influence of near-tree neighboring pairs; the resulting refined scores are in turn refined by reference distribution, which takes into account references in the regulations. The entire process together produces a reliable set of scores, and below threshold pairs of provisions are discarded as dissimilar pairs.

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 [14]. 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.

As in Chapter 3, most of what is proposed here has been completed. All the refinement techniques are implemented, with the base score computed using concept match. The

other proposed matches, e.g. index match, exception match, measurement match, section titles match and etc., have yet to be completed. Therefore, the performance of the system is assessed solely based upon the concept match.

4.2 Our similarity evaluation model

The comparison core takes two regulations and produces a similarity score, denoted by $f \in (0, 1)$, per pairs of provisions, e.g. pair (A, B) with section A from the ADAAG and B from the UFAS. The process starts with an initial similarity score which composes of scores from each of the features or evidences identified. Then the immediate surrounding nodes, in particular the parent, the siblings and the children of (A, B) are compared as well to modify their initial score. The influence of the not-so-immediate surroundings of nodes A and B is taken into account by a process called Reference Distribution. Here, based on the assumption that similar sections reference similar sections, the score between the referenced sections is used to update that of (A, B). After this process, a stable ranking of the most similar or related sections is produced.

4.2.1 The Base Score: f_0

The base score f_0 is a linear combination of the scores f_i from each of the features i. Scores from features can be weighted differently but for now equal weights are assigned to all features as in Equation 1. The scoring scheme for each of the features is discussed below.

$$f_0 = (\Sigma_{i = \text{features}} f_i) / \#\text{ features } i \quad (1)$$

4.2.1.1 Concept f_{cp} and Index f_{in} Match

Concepts are used exactly like the index terms in the vector model, and the regulations are indexed against these concepts. Each section is represented as a k-entry vector where k is the total number of concepts. Technique similar to the $tf \times idf$ measure is used for normalization, where term frequency (tf) is replaced by concept frequency, while the inverse document frequency (idf) remains the same. The formula to compute the idf component is taken to be $\log(n/n_i)$ where n is the total number of sections, and n_i is the number of sections the particular concept appears.

For two sections, the similarity score f_{cp} obtained by comparing concepts is given by the cosine similarity between the two concept vectors. Since cosine similarity is normalized, it always produces a score between 0 and 1.

Author-prescribed indices from the back of reference books or the regulation itself also follow the identical comparison procedure like Semio concepts. Indeed, it is interesting to see how human-written indices differ from machine-generated concepts with different weights in f_0. Again as mentioned above, we are keeping the same weights for all evidences for now.

4.2.1.2 Exception f_{ex} Match

If two sections share the same exception, it is intuitive that these two sections are more likely to be related. To compare exceptions, it is important to understand that exceptions are like sections with text, concepts, measurements and so on. Therefore, exceptions can be compared accordingly.

4.2.1.3 Measurement f_m Match

The similarity score f_m from measurement matching is assigned using similar techniques as in section 4.1.1.1, i.e. the $tf \times idf$ measure is used for normalization, where term frequency (*tf*) is replaced by measurement frequency, while the inverse document frequency (*idf*) remains the same. Thus two sections sharing a lot of unique measurements will get a very high similarity score, while two sections sharing no common measurements will get a zero.

4.2.1.4 Section Titles f_t Matching

Section titles always contain important key phrases central to the section; thus it is useful in identifying related sections. Section titles are typically very short, and are assumed to contain only one single idea. Therefore, it allows for a simpler computation of the similarity score f_t by taking the number of terms in the longest shared phrase across the two section titles, divided by the average lengths of the two section titles. Therefore two sections sharing the same title will have a score of 1, while two section titles sharing no common phrases will have a score of 0.

4.2.2 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 4.2.3.

In the score refinement process, we use the term psc collectively for the parent, siblings and children of a provision, i.e. its immediate neighbors. To help define the terms in a solid sense, we take sections A and B as our point of comparison. Thus to refine f_0(A, B), section A itself is first compared with psc(B), and vice versa, to produce the score $f_{s\text{-}psc}$. The next refinement takes into account the comparison between psc(A) and psc(B), which gives the score $f_{psc\text{-}psc}$. The final score f_{rd} comes from reference distribution, which compares the refereced sections. In addition, it is worth noting 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 $f_{psc\text{-}psc}$.

4.2.2.1 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, sections 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 have. We have

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

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.

4.2.2.2 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), which implies that another layer of indirection is inferred and thus the weight of psc-psc should be less than that of s-psc. We have

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

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)). Therefore the comparison threshold here is raised to $f_{s\text{-}psc}$.

4.2.2.3 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 materials as much. For instance,

sections in the ADAAG reference other sections in the ADAAG, but do not reference the UFAS or 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.

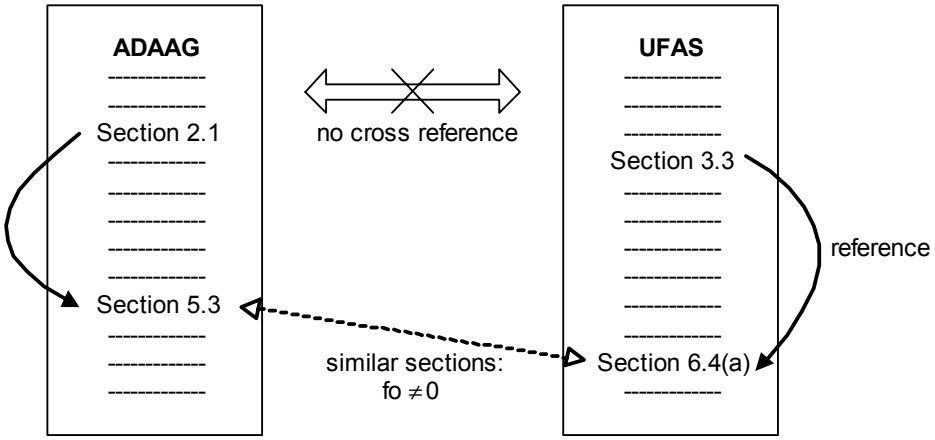


Figure 7: Comparison of section 2.1 from ADAAG with section 3.3 from UFAS

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 $f_{rd}(A, B) = f_{psc\text{-}psc}(A, B)$

4.2.3 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 just a prequel to conflict identification. Our assumption is, for two sections to conflict with each other, they must be related in the first place, and thus 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.

4.3 LSI: the control experiment

A traditional LSI approach [5] 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

$$a_{ij} = tf_{ij} \times \log (n / n_i) \tag{2}$$

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

$$[A] = [P][Q][R]^T \tag{3}$$

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

$$[A_s] = [P_s][Q_s][R_s]^T \tag{4}$$

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-to-document (doc-doc) similarity matrix is given by

$$[A_s]^T[A_s] = [R_s][Q_s]^2[R_s]^T \tag{5}$$

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 or UFAS against itself.

4.4 Sample Output

Currently the comparison core is a standalone application written in Java. The system will be transported to a web interface in the future as shown in Figure 5. Below in Figure 8, 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 as in Figure 9. With *compare*, the chosen pair of sections will be compared and ranked with all 5 scoring techniques as shown in Figure 10. The α's are parameters that the user can experiment with; all of them should be less than 1 as explained in the above sections.

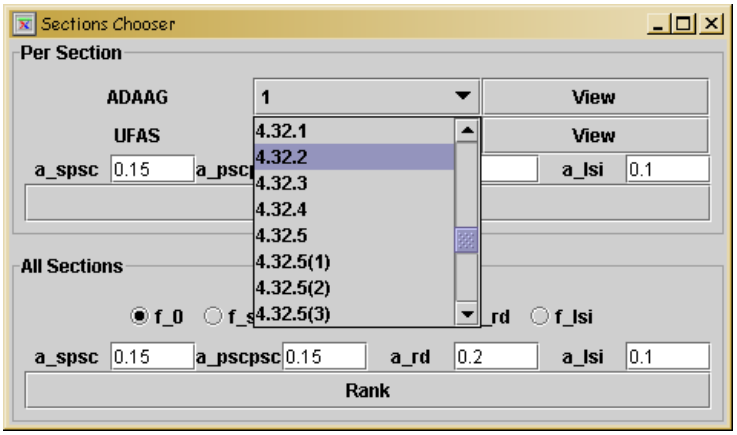


Figure 8: Graphical User Interface (GUI) of the comparison core

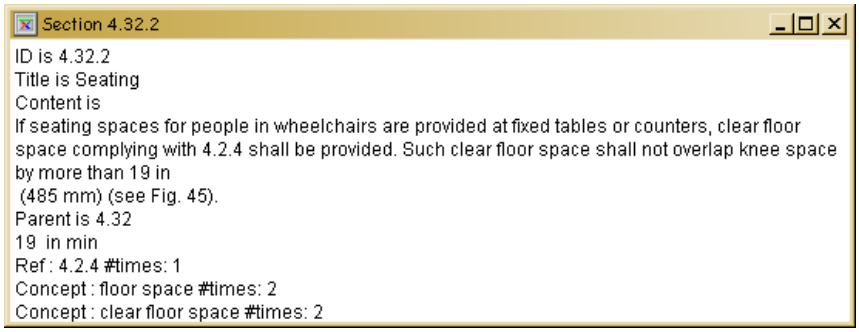


Figure 9: Body text of selected section

The lower half of the panel "All-section ranking" ranks and returns the top 20 sections according to the scoring technique the user chooses; please refer to Figure 10. Again the α's can be adjusted. Unless the α's are changed which forces an update of the

computation, a second *compare* or *rank* will not re-compute the results and thus should be immediate.

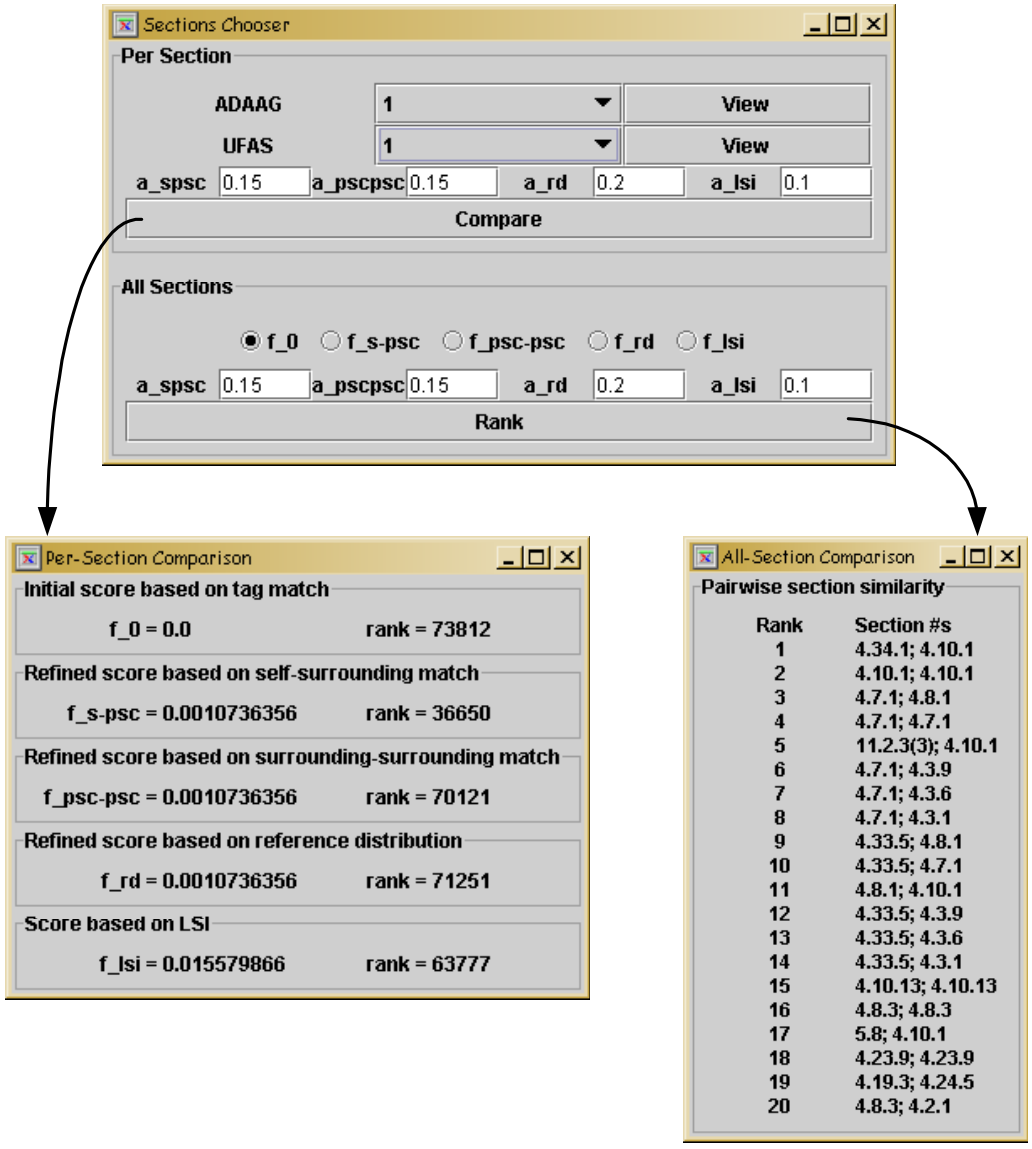


Figure 10: Compare and Rank

4.5 Brief Discussion of Implementation

The implementation of the scoring techniques is straightforward except for f_{lsi}. Indexing of terms is done by running over the data a first pass to build up the dictionary. A Java package Lucene [9] is used to parse the text with the provided lowercase filter, stopword filter and Porter stemmer. Then each section is checked against the dictionary to build a

word count vector. *Idf* values and vector normalization follow the word count (*tf*) process.

Our data is very sparse; after all the filters our term-document matrix is only 1% full. Thus data is written out as a sparse matrix to a file and Matlab is called in parallel with all other computations for f_0 to $f_{psc\text{-}psc}$. 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.

4.6 Discussions of Preliminary Results

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. In addition, the assessment of performance currently is based solely on the implemented features.

4.6.1 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, and in turn 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. We expect that this type of information can be captured with additional concept or domain-specific tags. Indeed it is clear that, from the performance of our system, we do need bag-of-word information, as well as noun phrases and structures to do a half decent job at comparison.

4.6.2 All-Section Ranking

All of the 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 pairs of sections with slightly different contents.

4.6.3 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.

4.6.4 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 are exactly the same.

4.7 Summary and Future Tasks

The comparison core 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.

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. 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.

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 base 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). Basically, all of the refinement techniques produce a refined score f_j 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 4.1.1 or 4.1.2.2, $[F_1]$ as the updated or refined similarity matrix after reference distribution, and α as the proportion of the update from references, we have

$$[F_1] = [F_o] + \alpha [R_1][F_o][R_2]^T \quad (6)$$

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 11 below.

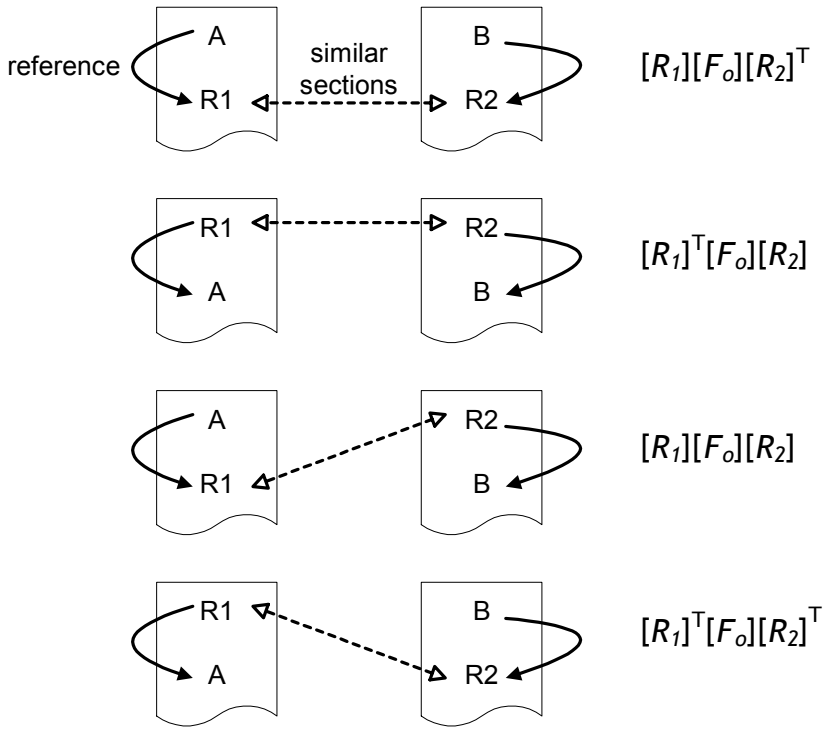


Figure 11: Comparison of sections A and B using f(R1, R2)

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.

5. Conflict Analysis

5.1 Introduction

In the motivation section of Chapter 1, some examples of conflicting provisions are given. These conflicting provisions are problematic since clients most likely will not know about their existences, or even if they do, they might not immediately recognize the provisions as conflicting. Thus tool to locate possible conflicting sections is helpful, and in our proposed system these sections are linked as showed in Figure 12. Note that here we assume that sections within a regulation will not conflict since they are released by the same agency, and thus there is no need to compare them.

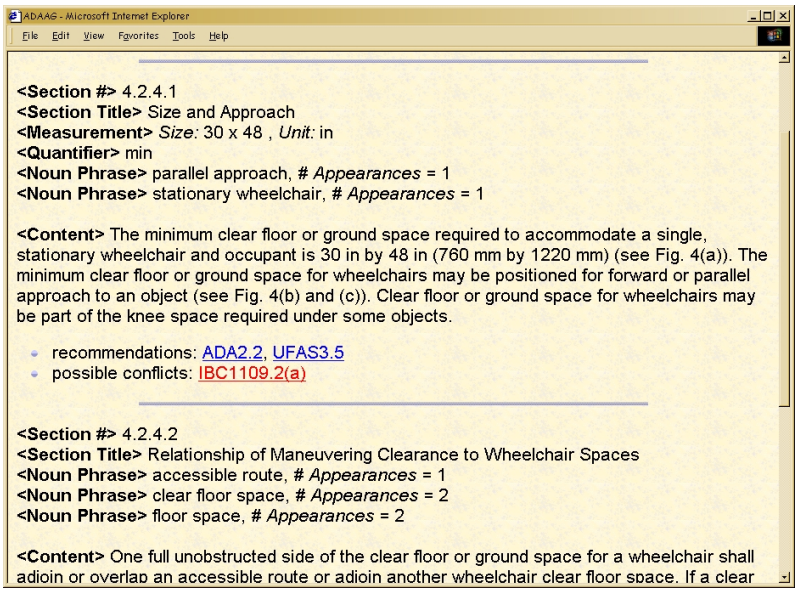


Figure 12: Possible conflicting sections linked to each provision

The conflict analysis core, which is showed on the right in Figure 13, starts upon receiving the identified list of related pairs from the similarity analysis core which is showed dimly on the left. The core takes only the most related pairs of provisions to check for conflicts, and the process starts with a part-of-speech tagger which tags a key term with its corresponding part-of-speech type. The result is passed to WordNet, an online lexical database for the English language, to consolidate the meaning of words, or in WordNet terms, the word *sense*. As a result, a list of senses for the key terms is produced, and the list is used to automate rule generation by our system. It also serves as an input to knowledge engineers to aid the rule writing process. Both machine-generated rules and human-generated rules are used to match the most related provisions, and a list of possible conflicting sections is returned.

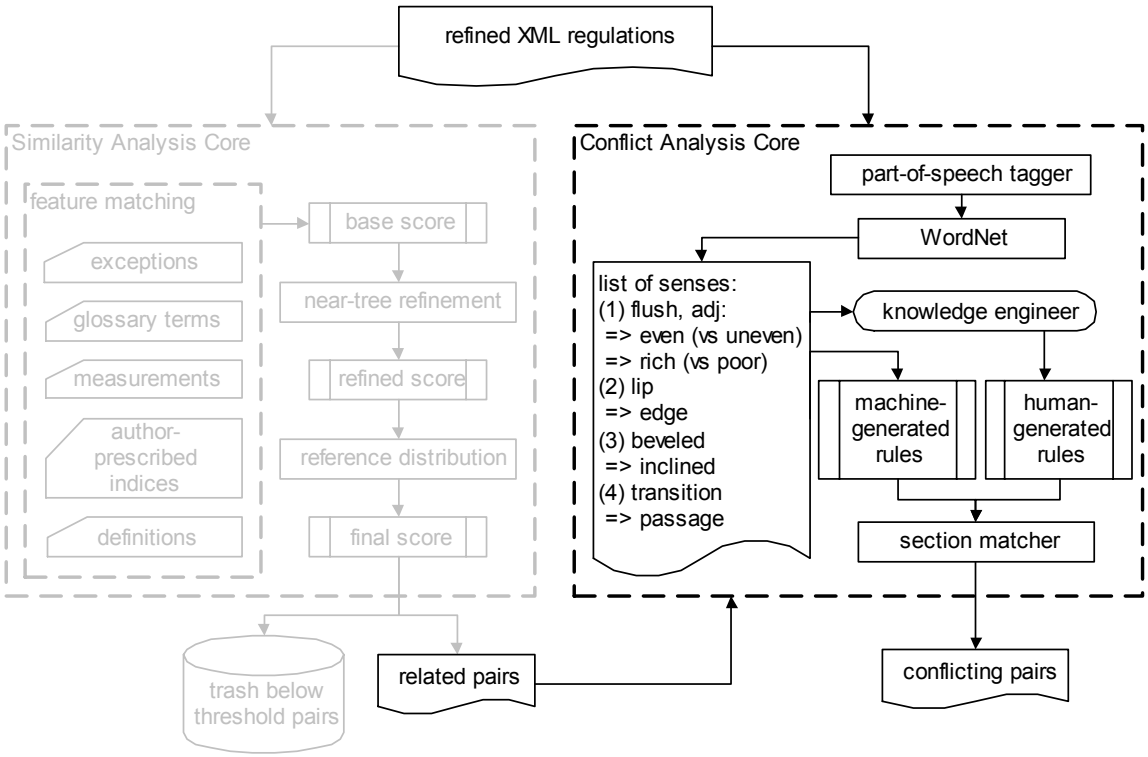


Figure 13: The Analysis Core Schematic

None of the proposed work here has been completed, and therefore it is still early to assess the performance of the conflict analysis core. We expect to use a public domain part-of-speech tagger with modifications to suit our need; interface to WordNet will be sought as well.

5.2 Part-Of-Speech Tagging

Different word type leads to different meaning even for the same word; for instance, the word *edge* shows the following senses as noun and verb in WordNet, a lexical database for English:

> The *noun* "edge" has 6 senses in WordNet.
> …
> 5. edge -- (a slight competitive advantage; "he had an edge on the competition")
> …
> The *verb* "edge" has 4 senses in WordNet.
> 1. edge, inch -- (advance slowly, as if by inches; "He edged towards the car")
> …

Clearly, the meanings are different. Therefore, to identify the meaning of a word in its context, its part-of-speech representation plays an important role. We propose to tag descriptive phrases, e.g. Semio concepts, definition and glossary terms, author-prescribed index terms, etc., with its part-of-speech representation in the section context. The major

part-of-speech types are used, namely nouns, adjectives, verbs and adverbs. Fine grain classification of part-of-speech types is not necessary since we will be using WordNet, which takes only these four types, to identify word senses. Indeed most of the time a distinction between a verb in its past participle form and a gerund does not give us extra understanding in its meaning, once it is identified as a verb.

5.3 WordNet

WordNet [15] is a lexical database for the English language developed at Princeton University. English nouns, verbs, adjectives and adverbs are organized into synonym sets (synsets), each representing one underlying lexical concept. Different relations link the synsets together, e.g. antonyms, hyponyms, etc. In our project, WordNet will be used to consolidate *senses* of words to serve as a shortlist to aid knowledge engineers to write domain-specific rules. In addition, antonymic information is used to flag possible conflicting ideas within sections.

Below is an example output from WordNet; the words are taken from the conflicting provisions discussed in Section 1.1. It takes as an input the interested word and its part-of-speech type, and returns a list of senses for the word where its synonyms can be identified. Shown on the right are the synonyms for the senses; in case of predicates, antonyms can also be identified and are listed in square brackets below.

flush, adj:	(1) even [vs. uneven], (2) rich [vs. poor]
lip, noun:	(1) edge, (2) …
beveled, adj:	inclined
transition, noun:	(1) passage, (2) …

5.4 Rule Generation

Rules are generated by our system to match conflicting sections; they can also be inputted by domain experts or knowledge engineers. In our domain, most of the conflicts are between descriptive phrases and measurements. Within-category conflicts, i.e. word-word conflicts and measurement-measurement conflicts, are classified as trivial conflicts which can be captured by machine-generated rules. Across-category conflicts, in our case word-measurement conflicts, are non-trivial and can only be extracted by human-written rules. As a result, our system minimizes human effort in rule writing; Figure 14 illustrates the idea.

Examples of machine-generated rules are trivial, e.g. WordNet antonyms may signal a rule of `conflict(flush, uneven)`, measurements can be compared numerically to identify conflicts, `<exception>` tags imply negation, and so on. On the other hand, human-written rules cover the non-trivial conflicts. For instance, the word flush clearly conflicts with any kind of slope measurements, thus the rule `conflict(flush, measurement(slope > 1:100))` can be an input from a knowledge engineer. However, it is simply impossible to generate a complete list of conflicting rules to cover all of the non-trivial conflicts, even for a domain expert with thorough understanding of the entire set of regulations. Thus, the consolidated senses serve as a shortlist to aid domain experts

to write rules that are not complete or exhaustive but nonetheless are enough to cover the domain.

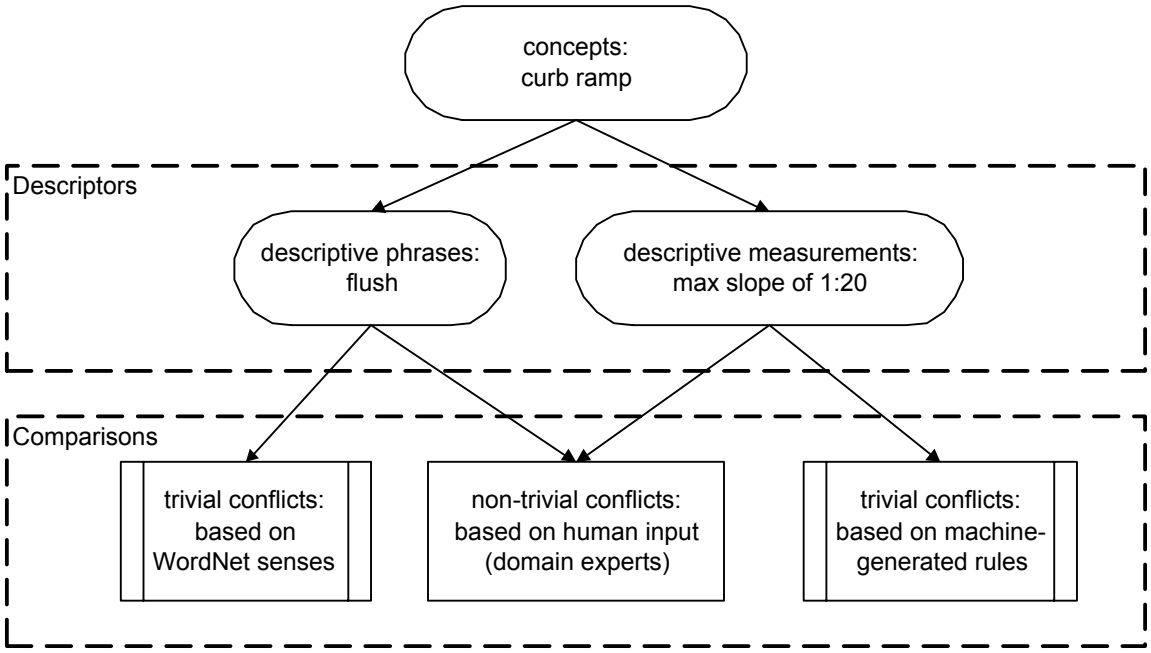


Figure 14: Matching rules coverage

5.5 Summary and Future Tasks

In this chapter, we briefly discuss different modules of the conflict analysis core. The core starts its analysis on the most related pairs of provisions as identified by similarity analysis, based on the assumption that conflicting sections are related over a certain topic area. The process starts by tagging descriptive phrases, for instance Semio concepts, with its part-of-speech type. This information helps to reveal the meaning of a word, or in linguistic language, its word *sense*. WordNet is used to consolidate word senses as well as to find conflicts between terms, and a list of senses is produced as a result. This list serves as an input to automated conflict rule generation by our system, which covers trivial conflicts between descriptive phrases based on antonym identification, and between measurements based on numerical comparisons. The list of senses also helps knowledge engineers to write conflict rule that covers non-trivial conflicts between descriptive phrases and measurements.

There are some issues to be solved in the future, for instance, WordNet is a useful tool to reveal lexical relationships between words but it is not an end solution to our problem. The vocabulary of the English language is large and currently WordNet contains about 50,000 synsets just for nouns. The vocabulary is not easily consolidated and comparison becomes unfeasible if a variety of similar-but-not-synonymous words is used in

regulations. Further investigation is needed to review the vocabulary size of typical regulation sets and to assess the performance of conflict analysis.

6. Discussion & Future Tasks

6.1 Discussion

This project aims to develop an infrastructure for regulation management and comparative analysis. A repository is built by transforming regulations into XML format because of its capability to handle semi-structured data. After all regulations are in a unified format, features, or evidences, are extracted from the set of regulations automatically or by a knowledge engineer, in addition to features from reference materials such as engineering handbooks. A taxonomy is built on top of the concepts identified by an information retrieval tool, such as Semio, to allow for easy viewing following the hierarchy.

Comparative analysis is divided into two tasks: similarity and conflict analysis. The similarity analysis core first computes a base score between pairs of provisions by combining similarity scores from each of the features. The base score is refined by taking into account neighboring sections. Reference distribution is performed to further refine the scores according to the reference structure in the regulations. A list of the most related sections is produced as a result.

After a set of reliable similarity scores is obtained, the conflict analysis core starts to identify possible conflicts among the most related sections, assuming that in the documents, sections must be related before conflicting ideas are presented on a certain subject area. A part-of-speech tagger is used to tag key terms with their part-of-speech types; then WordNet, a lexical database for English, is called to capture synonymic and antonymic relationships between terms. The list of senses from WordNet serves as both the input to automate conflict rules generation by our system and to aid knowledge engineers to write rules by hand. Both machine-generated rules and human-written rules are used to match possible conflicting provisions.

Some unsolved issues need to be dealt with during the course of system development. For instance, currently only out-references are incorporated in reference distribution; however in-references should also be compared to refine the similarity score between sections. Cross-references between regulations, and random references to outside domain in general are not considered due to the lack of such cases within ADAAG and UFAS. In addition, in conflict recognition, WordNet produces a rather large vocabulary and the *senses* of words are not truly consolidated, thus making comparison difficult.

6.2 Expected Contributions

The expected contributions from this study are valuable for the industry, particularly for small businesses. Small companies simply do not have the resources and cannot afford to hire lawyers or specialists to do compliance check for projects and developments, and thus often suffer from fines for regulation violations. The sheer volume of regulations from different governing bodies makes it difficult for small business to locate relevant information, which in turn hinders growth of such companies having to devote their

already-limited resources on either compliance check or budget for penalties. Therefore, an online regulation repository with browse and search functionalities could help small business to locate related provisions, and thus makes *understanding* of regulations easier. In addition, tools to group similar or conflicting provisions together significantly shortens the process of compliance check against the complicated set of regulations.

Apart from the practical applications of our system, the techniques developed in this project can be generalized to other domains as well. We proposed to compare regulations released by different agencies; however, the same regulation can be compared against itself, particularly in times of a new release of that regulation. Our tool can be used to find updates on the new version that differ from the old copy, which can be interesting as well.

There are plenty of text documents that are hierarchically structured, e.g. traditional textbooks are organized chapter by chapter, with sections and subsections within each chapter. Techniques that involve refinement of scores according to near-tree neighbors can be used to compare, in this case, a section in a textbook versus another. Reference distribution can be applied on other areas of document comparisons as well, and regulations are not the only highly referenced texts available. For instance, sections in software user manuals can be cross-linked as much as regulatory documents. Therefore, the algorithms developed in this project can be useful in general textual document comparison as well.

6.3 Project Schedule

Finally, as pointed out in previous chapters, some of the proposed work have yet to be implemented, and testing on other regulatory domain is expected after the completion of implementation to assess system performance. Below is a timetable listing each task, its expected completion time, and dependency on other tasks as shown by the vertical arrows. There are four main tasks in our project, namely the repository development, similarity analysis, conflict analysis and system testing; each of the tasks is composed of several subtasks.

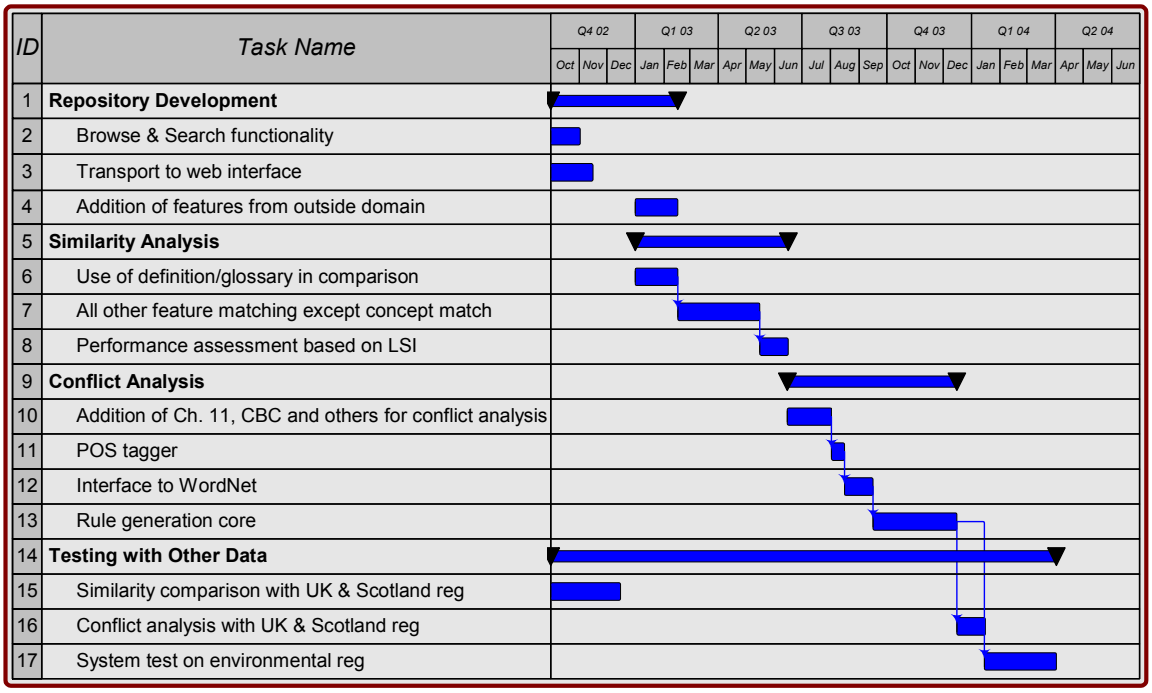


Figure 15: Schedule of project tasks

The repository development phase of our project is almost completed, and the only subtasks remain are to provide browse and search capabilities via a web interface, and also the addition of glossary terms and author-prescribed indices from engineering handbooks and references. The similarity analysis core is currently implemented with concept matching, score refinements and scores from LSI technique. Therefore, all other feature matching techniques apart from concept matching need to be completed. In addition, some theoretic work is listed as well, e.g. task 6 aims to produce the means to use definitions and glossary terms to help comparison, and task 8 explores possible methods to compare our scores with that obtained from LSI technique.

We expect the conflict analysis core to take longer than other tasks since none of the proposed elements is implemented. To start off, Chapter 11, Accessibility, of the CBC will be added to our corpus to illustrate conflicts since it is certain that conflicts exist between the CBC and the ADAAG, as shown in the motivation section of this proposal. A part-of-speech tagger, an interface to WordNet, and the rule generation and matching engine are sequentially implemented as they depend on one another. Finally, different system testing is performed at different stages of the project. A similarity comparison between our documents and the Scottish accessibility regulations is scheduled as task 15. Assessment of the conflict analysis core follows task 9, namely the conflict analysis, and a system wide testing on environmental regulation is scheduled at last.

A. Appendix*

A.1 Introduction

As mentioned in Chapter 3, a HTML to XML parser is developed for Federal regulations. It takes an HTML file as input and outputs the corresponding XML file, with added tags and removed HTML formatting. The first phase of implementation of the parser is to better understand the structure of the two sets of regulations, i.e. ADAAG and UFAS, so that the parser can automate as much as it can in the process of transformation. However, due to the relax format of HTML regulations, some manual editing before parsing is unavoidable, but it is kept to a minimum. The second phase is the actual development of the parser, and the tasks are further subdivided into developing different tags/elements. In XML, tags are usually called elements since they contain much more information than a regulation HTML tag, and in the following sections these two terms are used interchangeably. The final phase is to show the output of the XML files in a human readable form, and several figures are attached at the end of the Appendix.

A.2 Understanding the Basic Structure of the HTML Regulations

A lot of time is invested to understand the structure of the HTML regulations. After an initial examination of the regulations, it is clear that the HTML files for ADAAG and UFAS are well formed, meaning that tags are all ended properly. Browsers are very "forgiving" in parsing HTML in a sense that non-ending tags (and actually some tags are defined to be non-ending) are also accepted, while for XML the same browser will stop parsing and report an error. Therefore, it is very important for XML files to be well formed, and a lot of commercial products, e.g. HTML Tidy, are available to help to tidy up HTML files before transforming to XHTML or XML. These products might be useful in the later phase of the project when more HTML regulations are available, but not so for ADAAG and UFAS since they are already well formed.

As far as the structure of HTML regulations is concerned, ADAAG and UFAS provide nothing more than well-formed tags. The basic structure of the HTML code is as follows: sections or subsections start with the anchorage tag, or the target link tag `<a name=...>`, followed by the section name or title, closed with the ending `</a>` tag. The content of the target link tag is the section number, e.g. `<a name="2.1.4">`. In each section, paragraphs are distinguished by the paragraph tag `<p>`. Some definition tags, table tags (will be covered in later sections) and formatting tags are sprinkled in between sections.

At first glance, it might seem that the HTML document is readily parsable. However, since the regulations are not written for the purpose of parsing and transformation, the HTML format mentioned above is not always followed and is not consistent throughout the code. Here are three examples and the corresponding design choices:

* The parser is currently being re-designed by Ms. Pooja Trivedi and this does not reflect the latest XML structure.

(1) The target link tag does not always exist. For instance, in UFAS, the target link tag `<a name="1">` exists for section 1, but not so for section 2 or section 3. Thus the tags are manually added to the HTML file before running the parser.

(2) The content of the target link tag is not consistent; sometimes it refers to the section number, e.g. 2.1.4, while sometimes it refers to a randomly chosen name, which is usually taken as the first word of the section title. These are also manually corrected to reflect the section number for all target link tags.

(3) The target link tag does not always end at a predefined position. For instance, in ADAAG, some section headings have `<a name=…> Section Heading or Name </a>` while some others have `<a name=…> </a> Section Heading or Name`. Since this inconsistency happens much more frequently than that of (1) or (2), it makes more sense to let the parser do the job rather than human brute force correction.

After correcting some minor inconsistencies, all other tags in the HTML files are examined. There are a lot of formatting tags, e.g. the bold tag `<b>`, the italic tag `<i>` in the file, which are not needed in the XML version of the regulation since XML separates content (XML) from format (XSL). This leads to the first job of the parser, preprocessing.

A.3 Development of the parser

The parser takes in a well-formed HTML regulation file and reads in the text. Then it starts preprocessing the text by removing useless tags. After all is done, it starts a recursive process of section element extraction and writes out the resulting XML text during the process.

A.3.1 Preprocessing

As stated in the above section, the HTML regulations contain a lot of formatting tags that are not useful in XML. Therefore, these tags are removed except the paragraph tag `<p>` which helps to subdivide the contents of a section into paragraphs; please see Table A1 below for the list of removed tags. Multiple spaces are collapsed into a single space, and the hyperlink tag `<a href=...>` is removed to ease parsing. In addition, the commercial product Semio Tagger helps to build the linkage between sections and concepts, and thus the hard coded hyperlinks present in the HTML files are no longer needed.

Removed HTML tags	Meaning
h1, h2, …	Headings
br	Line break
b	Bold font
i	Italic font
blockquote	Indented quotes

Table 1: Removed HTML formatting tags

A.3.2 Recursive development of the section tree

After preprocessing the text, the parser starts parsing from the beginning and consumes characters till the end of the buffer. Here an understanding of the structure of the regulation is helpful: the regulation is structured by sections, followed by subsections and so on. Each section has unlimited number of subsections (horizontally), and the tree depth (vertically) is unbounded. It is best illustrated with a figure:

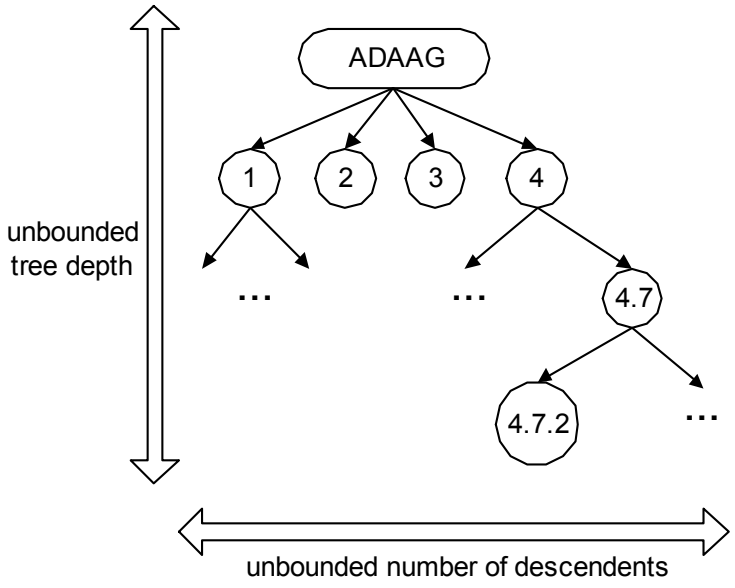


Figure 16: Regulation Structure

Since there is no predefined tree depth or number of descendents for a section, a recursive parsing algorithm is best to capture the hierarchical structure. To implement this functionality, the parser goes on a while loop and calls CreateSectionElement (Please see Figure A2 and A3) to create the top level of the section elements (sections 1, 2 and etc.; denoted by level 1).

A.3.3 Section Tag

Section Tags are created by the method CreateSectionElement. A section tag contains the following information as attributes:

(1) *ID*: represents the section number. For instance, 4.1.6(1)(a) denotes a subsection where . and ( ) are used as delimiters for section levels.
(2) *Name*: represents the section title or heading.
(3) *Asterisk*: denotes whether that particular section has related material in the Appendix. In the HTML version of the regulation an asterisk is appended to the section title for reference; here using XML this information can be extracted as metadata stored in an attribute. 0 represents no while 1 represents yes.

Combining the above attributes, a typical section tag looks like `<Section ID="4.1.6(1)(a)" name="Wheelchair Passage Width" asterisk="1">`. To extract the ID of each section, since we assume that all sections have been preprocessed with the tagging `<a name=…>`, the content of the name attribute in the original HTML file is taken as the ID field of the new XML section tag. As mentioned above, due to the inconsistency of the placement of the section title in the original HTML code, the name field of the XML section tag is taken to be either the content enclosed by the original `<a>` tag, or what follows the ending `</a>` tag terminated by a period. It is easiest to understand with a figure:

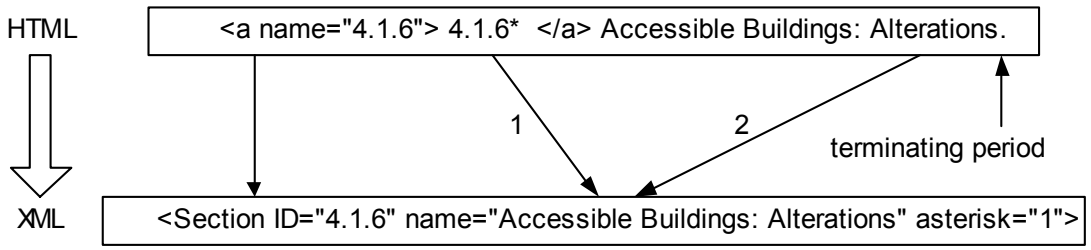


Figure 17: Parsing the metadata of each section

In addition to the above rules of parsing, after examining the ADAAG and UFAS regulations further, it becomes clear that starting from the 4th layer onward there is no section name. For example, sections 4.1.2(1) or 4.1.2(1)(a) do not have a section title. In these cases, the name field is empty; i.e. `<Section ID="..." name="" asterisk="...">`.

After parsing the attributes for the section tag, the parser starts to extract content out of the section. A section can contain one of the following three contents:

(1) End of file (EOF). No more character is found.
(2) Another Tag
    a. `<a>` tag – another section; may or may not be a child of the current section
    b. `<p>` tag – a paragraph belonging to the current section
    c. `<dl>` tag – a definition belonging to the current section
    d. `<table>` tag – a table belonging to the current section
(3) Plain text

(1) and (3) are easy; the parser just writes the characters (or nothing) as element content of the section tag, followed by a closing `</Section>` tag. All tags in (2) are easy to handle except (a); for (b) to (d) since they are a child element of the current section element, we simply call CreateParagraphElement or CreateDefinitionElement respectively. Please see sections A3.4 and A3.5 for details and Figure 18 for a schematic of the function calls.

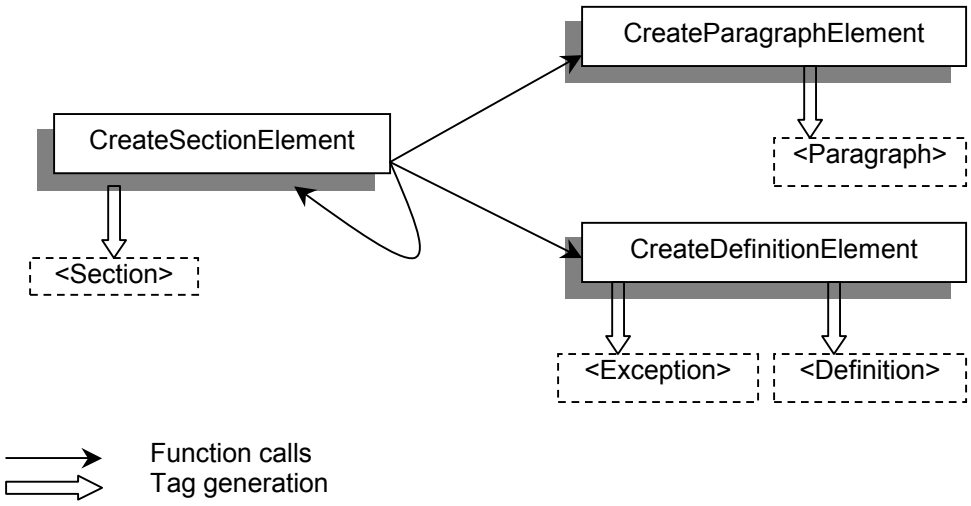


Figure 18: Schematic of function calls in building the regulation tree

For a `<a>` tag in the section content, we have to compare this new `<a>` section with our current section to check to see if it is a descendent of the current section or not. This is done by splitting the section ID with the delimiters . and ( ); for instance 4.2.1(1)(a) is split into [4 2 1 1 a]. After splitting and comparing, we have one of the following three scenarios:

(1) The new section ID is shorter than or equal to the length of the current section ID, e.g. new ID = [4 2 2] and current ID = [4 2 1 1].
    ⇒ The new section is not a descendent of the current section.
(2) The new ID is longer than the current ID:

a. The new ID's 2nd to the last digit is the same as the last digit of the current ID, e.g. new ID = [4 2 1] and current ID = [4 2]
   ⇒ The new section is a descendent of the current section.
b. The new ID's 2nd to the last digit is not the same as the last digit of the current ID, e.g. new ID = [4 3 1] and current ID = [4 2]
   ⇒ The new section is not a descendent of the current section.

For cases (1) and (2)(b), the new section is not a child of the current section, thus the parser ends the current section with `</Section>` and calls CreateSectionElement recursively on the new section found. For case (2)(a), since the new section is a child of the current section, the parser calls CreateSectionElement on the new section before ending the current section. This process is recursive due to the nature of the regulation tree.

A.3.4 Paragraph Tag

Paragraph Tags are created by the method CreateParagraphElement, which is called upon parsing of a `<p>` tag. Based on the assumption that in the original HTML file the `<p>` tags are ended properly with a `</p>` tag (which can be enforced by running a tidy-up program before parsing; in our case both ADAAG and UFAS are well formed and the assumption is justified), a paragraph element is created with contents up to the ending `</p>` tag. A typical paragraph element looks like `<Paragraph> actual content of the paragraph </Paragraph>`.

A.3.5 Definition Tag

Definition Tags are created by the method CreateDefinitionElement, which is called upon parsing of a `<dl>` tag. A `<dl>` tag in HTML represents a phrase and its corresponding definition, which is usually indented in browsers. For some mysterious reason (again, the HTML regulations are not intended to be transformed into XML), the ADAAG and UFAS use `<dl>` tags to represent both normal definitions and exceptions, which complicate the parsing of a `<dl>` tag since it does not just represent a definition. Therefore, the parser first checks whether the definition tag actually contains a pair [phrase, definition] (e.g. [Access Aisle, An accessible pedestrian space …]), or a pair of exception [section content, exception content] (e.g. [… parking space…, EXCEPTION: Provision for all required parking…]). If it contains the exception as a "definition", it creates an Exception tag instead of a Definition tag. An exception tag simply contains the content of the exception in the form of `<Exception> exception content </Exception>`.

If the `<dl>` tag represents an actual definition, a definition tag is created with 3 attributes:

(1) *Term*: the term being defined.
(2) *Definition*: the definition.
(3) *Asterisk*: as in a section element, represents whether relating material exists in the Appendix or not.

A complete definition tag looks like `<Definition term=”Access Aisle” definition=”An accessible pedestrian space...” asterisk=”0”>`.

A.4 Outputs

Without XSL, the default XML display in a commercial browser looks like a parse tree, which is not at all easy to view. However, it does give the basic layout of the regulation and the metadata stored in the attributes. Figure 19 shows the output of the UFAS XML regulation displayed with Internet Explorer 5 (IE5).

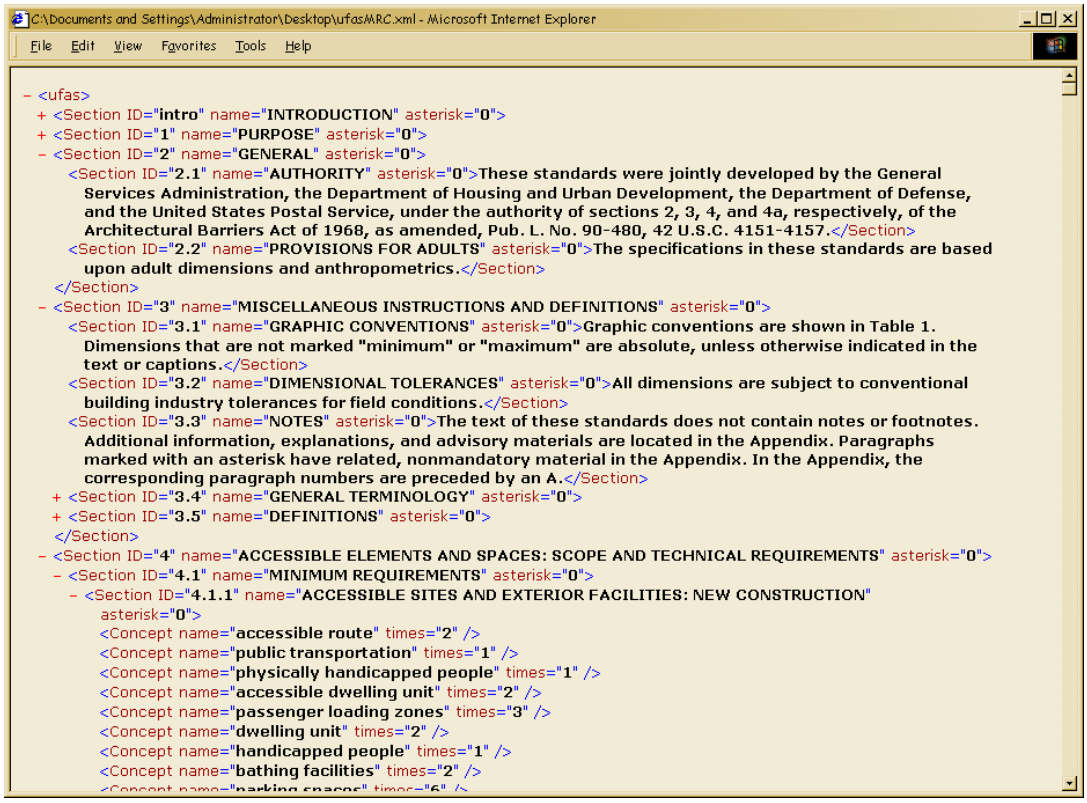


Figure 19: UFAS in XML displayed with IE5

Figure 20 below is rendered in Microsoft XML Notepad, a freeware to aid editing in XML. It shows clearly the tags in each provision and the section hierarchy. Figure 21 shows a sample definition tag in ADAAG, where the name and definition of a phrase is stored as attributes in the definition tag.

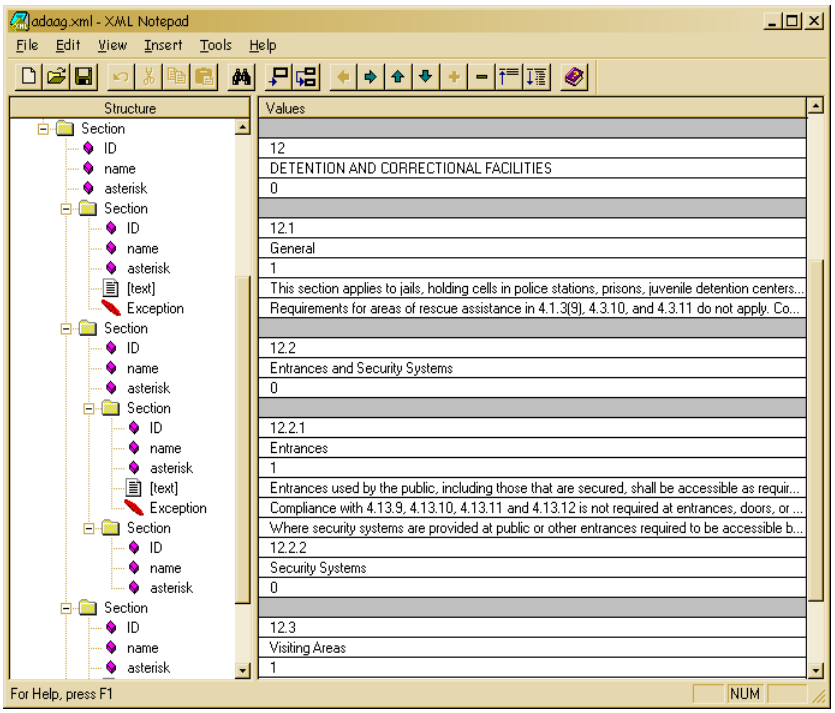


Figure 20: ADAAG in XML displayed using Microsoft XML Notepad

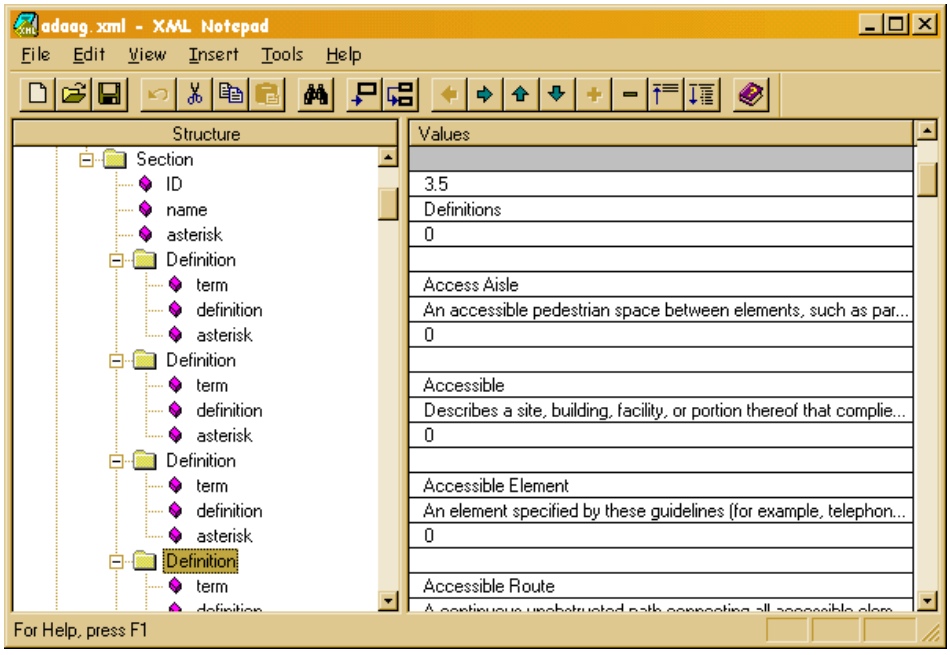


Figure 21: Definition Tags in ADAAG

Bibliography

1. The Access Board, 1998. *ADA Accessibility Guidelines for Buildings and Facilities*.
2. The Access Board, 1986. *Uniform Federal Accessibility Standards (UFAS)*.
3. Baeza-Yates, R., and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. ACM Press, New York, NY.
4. Bishop, C. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press; Clarendon Press, New York, NY.
5. Deerwester, S., et al. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 391-407.
6. Dorre, J., Gerstl, P., and Seiffert, R. 1999. Text Mining: Finding Nuggets in Mountains of Textual Data. In the *Proceedings of The ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 398-401.
7. Gibbens, M.P. 2000. *California Disabled Accessibility Guidebook 2000*. Builder's Book, Inc., Canoga Park, CA.
8. International Building Code 2000, 2000. *International Conference of Building Officials*.
9. The Jakarta Project, 2002. *Jakarta Lucene*. http://jakarta.apache.org/lucene.
10. James H. Garrett, J., and Liang, V.-C. 1998. *A Wed-Based Broker for Regulations*. Institute for Complex Engineered Systems, Pittsburgh, PA.
11. Jing, H., and Tzoukermann, E. 1999. Information Retrieval Based on Context Distance and Morphology. In the *Proceedings of 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, 90-96.
12. Jones, K., and Willett, P. 1997. *Readings in Information Retrieval*. Morgan Kaufmann, San Francisco, CA.
13. Kidder, F., and Parker, H. 1931. *Kidder-Parker Architects' and Builder's Handbook*. John Willey & Sons, Inc., London, UK.
14. Manning, C.D., and Schutze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.
15. Miller, G.A., et al. 1993. *Five papers on WordNet*. Cognitive Science Laboratory, Princeton, NJ.
16. Mitra, P., and Wiederhold, G. 2001. An Algebra for Semantic Interoperability of Information Sources. In the *Proceedings of 2nd IEEE Symposium on BioInformatics and Bioengineering*, Bethesda, MD, 174-182.
17. Porter, M.F. 1980. An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 130-137.
18. Semio Corporation, 2002. *Semio Tagger*. http://www.semio.com.
19. Zeleznikow, J., and Hunter, D. 1994. *Building Intelligent Legal Information Systems*. Kluwer Law and Taxation Publishers, Deventer, the Netherlands.