Characteristics of Document Similarity Measures for Compliance Analysis

Asad Sayeed
Dept of Computer Science
University of Maryland
College Park
MD 20742 USA
asayeed@cs.umd.edu

Soumitra Sarkar
IBM Research
P.O. Box 704
Yorktown Heights
NY 10598 USA
sarkar@us.ibm.com

Yu Deng
IBM Research
P.O. Box 704
Yorktown Heights
NY 10598 USA
dengy@us.ibm.com

Rafah Hosn
IBM Research
P.O. Box 704
Yorktown Heights
NY 10598 USA
rhosn@us.ibm.com

Ruchi Mahindru
IBM Research
P.O. Box 704
Yorktown Heights
NY 10598 USA
rmahindr@us.ibm.com

Nithya Rajamani
Collaboratory for
Service Science
IBM Research - India
Hyderabad
AP 500032 India
nitrajam@in.ibm.com

ABSTRACT

Due to increased competition in the IT Services business, improving quality, reducing costs and shortening schedules has become extremely important. A key strategy being adopted for achieving these goals is the use of an asset-based approach to service delivery, where standard reusable components developed by domain experts are minimally modified for each customer instead of creating custom solutions. One example of this approach is the use of contract templates, one for each type of service offered. A compliance checking system that measures how well actual contracts adhere to standard templates is critical for ensuring the success of such an approach. This paper describes the use of document similarity measures - Cosine similarity and Latent Semantic Indexing - to identify the top candidate templates on which a more detailed (and expensive) compliance analysis can be performed. Comparison of results of using the different methods are presented.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Compliance analysis, Contract template retrieval, Cosine similarity, Latent semantic indexing, Text mining

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'09, November 2–6, 2009, Hong Kong, China.
Copyright 2009 ACM 978-1-60558-512-3/09/11 ...$10.00.

1. INTRODUCTION

In recent years, the IT Services business has shifted from being a captive market controlled by a few large organizations to a highly competitive one where many players of various sizes compete. As a direct consequence of this shift, improved quality, lower costs and shorter delivery time have become essential for maintaining and/or increasing market share. Today, IT companies are seeking to reduce the cost and time of service delivery by moving from a labor-based to an asset-based approach. The centerpiece of an asset-based approach is the development of standard reusable components created by domain experts which can be easily tailored to a customer's requirements, as opposed to developing custom solutions that are created "from scratch" for each customer by practitioners with varying degrees of expertise.

One example of standardized assets that is being increasingly adopted in the IT services industry is in the area of contract authoring. In that space, two key elements are necessary to adopt an asset-based approach. The first is the creation of standard service contracts, optimized for each type of service (e.g., setting up a Voice-over-IP service for a customer, or setting up a server cluster with site failure-recovery capabilities), which practitioners can reuse with minor modifications across clients. The second is a governance process that ensures the use of such standard contracts in the field.

Contract standardization is achieved with the use of *templates*. For a specific service, a contract template typically standardizes the title, the headings of all sections such as the Scope of Work, Service Provider Responsibilities, Terms and Conditions, Asset (Parts) List, and the text of each section – with appropriate placeholders for customization. A template also typically defines a standardized work breakdown structure, describing how the service will (ideally) be delivered in terms of the sequence of activities and tasks (some optional) to be performed. Such a work breakdown structure is typically described in a section of the contract that covers the service provider's responsibilities.

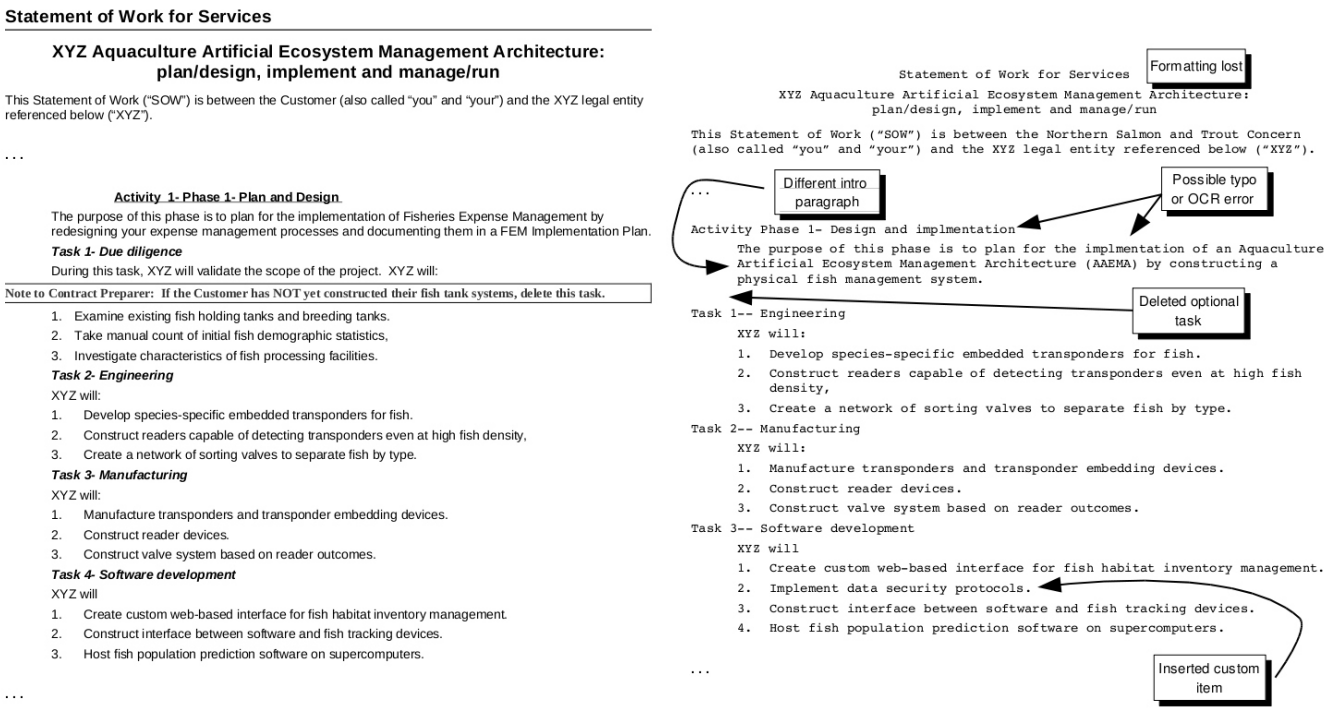

Figure 1: Contract template (*left*) and instance (*right*)

A critical component of such a standardization effort is a compliance checking system that measures how well the actual contracts that are being written in the field adhere to standard templates. This paper outlines the architecture of a contract template compliance checking system, and then focuses on text mining techniques used. These techniques are used to implement a front-end component that identifies, for each contract, the top candidate templates on which a more detailed structural and content analysis can be performed to measure compliance accurately.

Contract template compliance can be represented as a document classification problem. There is a large body of literature on document classification [1]. For this problem, each template can be considered to be a class. Some techniques such as Support Vector Machines (SVMs) and Naïve Bayes are often used for text classification [2, 3], but these are binary classifiers. As we have multiple templates, we will need to train multi-class classifiers. It is possible to turn binary classifiers into multi-class classifiers by training a set of binary classifiers, one for each class [4]. However, this is impractical, since the number of templates varies over time; every time a template is added, ground truth must be generated, and at least one new classifier must be trained. Consequently, we chose to treat this as an information retrieval (IR) problem, with the contract documents acting as queries on a database of templates. To apply IR techniques, we chose to exploit a vector-space model of document similarity.

In this paper, we present work based on existing IR-based document classification and clustering efforts, particularly related to Latent Semantic Indexing [5]. We apply this to document instances that are expected to correspond to templates, but for which the relationship may be less than direct. The task of contract preparers includes some degree of customization of both the structure and the content of a template before it becomes a contract acceptable to a customer. Secondly, a subset of the documents have noise introduced by the OCR process (since the contracts are stored as images) which may need to be smoothed out. The unique characteristics of this problem suggest starting with an approach that does not highly weight structural information and can involve a measure of semantic relatedness. Thus, we chose to compare the use of a vector-space model with Latent Semantic Indexing to one without. This approach was chosen because it is simple to implement, customize and experiment with.

2. THE SERVICE CONTRACT TEMPLATE COMPLIANCE SYSTEM

2.1 Templates and Instances

There exists an extensive literature in artificial intelligence techniques for reasoning about business contracts. Techniques in this area range from defining formal languages and logical systems in contract representation [6] to integrating legal knowledge into expert systems that govern contract formation [7].

In these cases, contracts are represented in a formal, structured manner. This is *not* the case for the types of contract templates and instances we discuss here. For our purposes, contract templates are loosely structured documents in human language developed by engineering, sales, and legal teams. Templates are generated or removed from the library that contains them whenever the organization changes the services offered to customers. This can happen fairly fre-

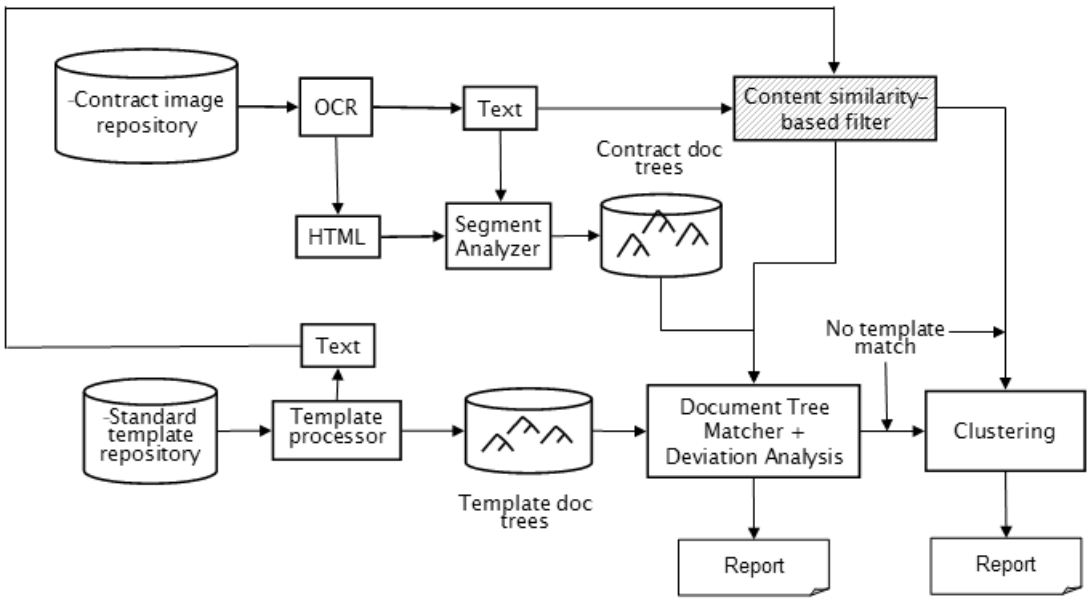

Figure 2: Architecture of SOW Compliance System

quently depending on the overall dynamics of the business environment.

Sales teams generate instances of templates as they deal with customers. Contract templates are often intended to be fairly flexible in their customization as instances. Not only do they have spaces for client names, dates, and so on, they also have short removable segments and spaces to add more detailed instructions at the client's request. In fact, the entire document can be edited at will; while the goal of the exercise is to maximize compliance with the template, there is some flexibility required for sales teams to conduct business. In our case, the instances are generally stored as fax images of signed documents rather than text. Figure 1 contains an example template and instance pair for aquaculture management services provided by a fictional company; the instance contains a few examples of the alterations that can happen when a template is instantiated, edited, and then potentially stored as a fax image.

The wide variation in contract structure, the potentially large number of contracts, and the potential for flexibility in contract instance generation suggest that we use document similarity and classification approaches; the remainder of this work describes our own approach.

2.2 System Architecture

Figure 2 describes the architecture of the overall compliance checking system. The input to the system is a repository of contract images stored in Portable Document Format (PDF), and standard templates which are Microsoft Word documents. The output is two sets of reports, one listing contracts that do not match any templates, collated in groups based on similarity of the members, and the other showing, on a per template basis, the percentage of matches, the trend of deviations, and corresponding revenue and profit information.

Contract images are processed by an Optical Character Reader (OCR) tool which generates plain text and HTML output. The Segment Analyzer attempts to recover the document structure from the text and HTML inputs. It uses a combination of statistical techniques (language models) and heuristic rules to construct the document tree.

Contracts are next processed by the Content Similarity-Based Filter, which is the subject of this paper. This module identifies the top candidate templates which should be considered for detailed (and more expensive) tree structure-based matching.

The Document Tree Matcher compares a contract tree to a template tree using tree matching algorithms, identifies the mismatches (deviations), and computes a combined structure-content similarity score for the two documents. The Deviation Analysis step analyzes the differences between a contract and the best matching template (if one exists) for each set of contracts associated with a given template, to produce insights on the types of customizations being made to specific templates (for specific service types) when creating service contracts from them.

The Clustering module attempts to group together similar contracts, ideally those dealing with the same service type, in order to identify new types of services for which standardized termplates might need to be developed.

2.3 Technical Challenges

2.3.1 The Contract Template Compliance Problem

The eventual determination of whether a service contract is compliant with a given template has to be made using a combination of structural and content-based similarity analysis. A contract can be compliant with a template even if there is not a perfect alignment of section headings, nor a high degree of similarity between the contents of contract and template sections that align with each other based on

heading text. Such differences can be present due to various reasons:

- An archived contract could have been created from an older version of the template.
- Sections at a certain level might have been reordered to change the emphasis on certain aspects of the contract based on customer priorities expressed in the requirements.
- The content of each section can be more than trivially different from that of the matching section of the template. Certain details in the template, such as steps in a work breakdown structure identified for service delivery, may not make sense due to some unique considerations in the customer environment and thus have to be omitted.

In fact, the last point is the very reason why the compliance system architecture includes a deviation analysis module; to understand if certain aspects of the template are being modified frequently enough to warrant another look at its adequacy to address current market demands.

Computation of a matching score that can take into account the types of differences outlined above between a contract and a template requires matching a contract with a template at a structural (document tree) level, followed by measuring similarity of section contents. Aligning contract and template document trees, while accounting for section and section level reordering and section heading differences, is computationally expensive.

In a large service organization, there can be a dozen or so major domains in which services are offered, such as servers, storage, networking, mainframe, data center management, etc. For each area, dozens of offerings may be available. For example, in storage services, typical offerings can include storage environment optimization analysis, storage virtualization setup, disaster recovery readiness analysis and setup, installation and configuration services for specific types of storage devices - which tend to be complex to set up optimally, storage area network configuration with fiber channel equipment, etc. Therefore, it would not be unreasonable to assume that there can be several hundred templates that span the range of service offerings.

Performing tree matching of a specific service contract against several hundred templates to find the best match (or to determine that there is none) is very inefficient. Thus it is important to incorporate a front end filter that can identify, for each contract, the best candidate templates on which tree matching should be performed. The Content Similarity-based Filter component performs that function and is the subject of this paper. The challenges of performing such filtering using document-level content similarity measures are outlined in the next section.

2.3.2 *Challenges of Similarity-Based Template Filtering*

In order to identify the top candidate templates for detailed matching, the Content Similarity-based Filter component uses document-level similarity measures, specifically *Cosine Similarity* as well as *Latent Semantic Indexing*. The key technical difficulty in precisely identifying the top candidate templates for each contract is due to the mixed quality of the contracts.

Since the system is designed to perform compliance analysis on historical data, the quality of archival data is a major concern. To ensure ease of electronic access many years into the future, archival stores typically retain documents as images that are scanned and converted into PDF files. Image quality is highly dependent on the quality of the scanner and its settings.

To perform any kind of text mining on images, they have to be processed by an OCR tool. While currently available OCR tools are quite advanced in their ability to convert document images to text (and HTML or equivalent) files, images scanned with low contrast or high page misalignment (rotation) typically result in a lot of errors in the resulting text document. Such errors appear in the form of missing letters in words, missing spaces leading to concatenated words, missing character sequences spanning multiple words, introduction of spurious characters, and original line boundaries not being preserved in the text file. Many of these classes of errors lead to textual differences between a contract and the template it is supposed to match, which can be challenging for text-based similarity measures to overcome.

Additionally, deliberately introduced deviations in a contract derived from a template, for reasons outlined in Section 2.3.1, can also confuse a document-level similarity measure. Furthermore, a contract can differ from a template due to expected customizations that represent a specific customer environment. For example, a typical service contract may have an appendix where the Inventory (Parts) List is to be inserted. Such a list, for even a given service type, can vary greatly from one customer to another. One customer might order a single piece of equipment whereas for a larger contract for the same service, dozens of parts might be ordered.

Finally, it was observed that due to inconsistencies in policies, certain contract images include the cover letter, the signature page and other small appendages whereas others do not - thereby making a document-level similarity match between a contract and a template inherently inaccurate if such appendages are considered to be part of the document.

The initial design of the content similarity-based filter ignores all these sources of errors, since attempting to eliminate them would be counter to the goal of building a highly efficient filter. Experiments were conducted using entire contract documents generated by the OCR tool, and no attempt was made to preprocess the OCR output to correct spelling errors or detect spurious sections to eliminate before measuring document level similarity.

Further details of this component are provided in Section 3.

3. IMPLEMENTATION

The internal structure of the content-similarity based template filtering system is shown in Figure 3. The system is prototyped in Java, using Apache Lucene to index the contract as well as the template documents after conversion to text using a commercial OCR tool. Two different techniques are used to compare contracts to templates to identify top candidate templates for more detailed analysis. Each technique depends on a term vector representation of a document. In one case, *cosine similarity* is used, whereas in the other case *Latent Semantic Indexing* is used for dimensionality reduction before applying cosine similarity. Details about these methods, referred to in the rest of the paper as

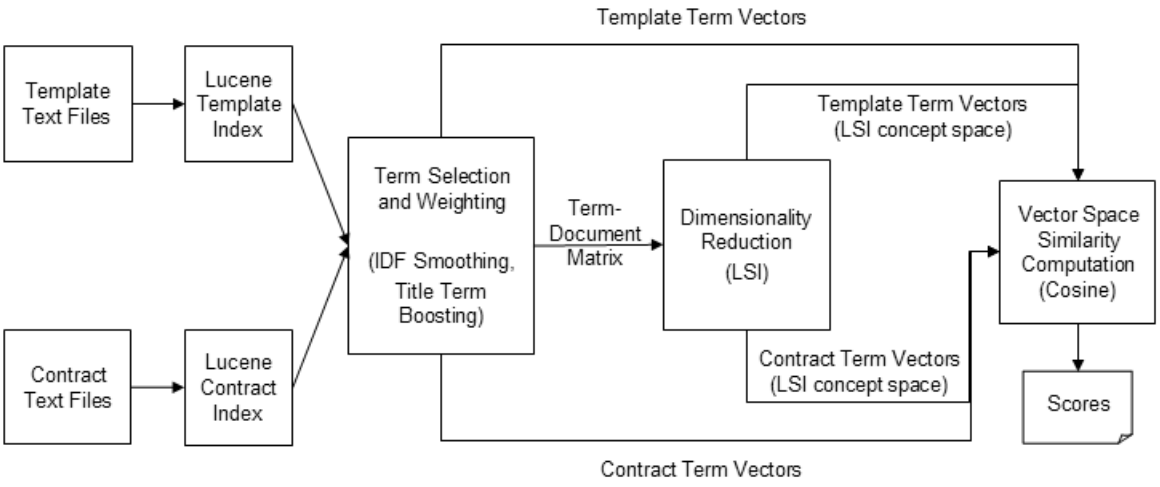

Figure 3: Document Content Similarity-Based Template Filtering

cosine similarity and LSI, respectively, is provided in the next section.

3.1 Document similarity

Document similarity is computed using a vector-space model. In this kind of model, each document is conceptually viewed as a list of words, and is considered to be a member of a collection of documents with n types of words (as opposed to tokens, which are individual instances of words). Next, word counts are collected, and for each document, a vector is constructed of the form $(w_1, w_2, ..., w_n)$, where n is the total number of terms in the entire collection of documents, and w_n is the word token count of the nth word in the collection-wide list of word types. This defines a space of dimensionality n and represents each document as a vector in that space. Note that typically, certain common words (known as "stop words") such as "and", "if", "but", etc. are ignored when selecting which words should form the dimensions of the vector.

The similarity between two documents can be measured in terms of the length-normalized distance between vectors. One such measure is known as *cosine similarity*. Let $\vec{v}$ and $\vec{w}$ be document vectors of dimensionality n. Then the cosine of the angle between them is:

$$sim(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\|\|\vec{w}\|}$$

One technique in information retrival systems is to treat queries as documents. Then they can be converted into vectors of the same dimensionality as the document vectors in the database. Retrieval can proceed based on comparison between query vectors and document vectors.

In the above model, there are at least two major sources of bias in the documents retrieved. One of these is the fact that longer documents with more occurrences of a given term will obtain higher similarity scores with a given query containing that term than shorter documents. Thus, for term i in document j, normalization is done by converting the count $n_{i,j}$ into a term frequency (tf).

$$tf(i,j) = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Another source of bias is the set of terms that are very frequent in the collection. Their distribution among many documents suggests that they are unimportant and risks masking distinctions made by less frequent, more significant terms. To compensate for this, the inverse document frequency (idf) of every term is computed. If D is the set of documents, and t_i represents the ith term in the entire set of terms in the collection, then

$$idf(i) = \log \frac{|D|}{|\{d : d \in D \ \& \ t_i \in d\}|}$$

This divides the total number of documents in the collection by the number of documents in which t_i appears. Terms which appear in more documents will have an idf that is closer to 0.

Such $tfidf$-reweighted vectors can be obtained by replacing all term counts in each document vector by the result of multiplying tf and idf.

For contract-template compliance analysis, each contract is treated as a query on a database of templates. Both query and document are *tfidf*-reweighted vectors.

3.2 Dimensionality reduction

One problem with computing similarity in an n-dimensional space, where n is the number of (non-stop-word) terms in the collection of documents, is that there can be too many dimensions. This can detract from the task of capturing important similarities. This is reflected in the problem of *synonymy*. Relative to a given application of document similarity, it is conceivable that there is no interesting difference between "seat" and "chair". However, each of these will represent a different dimension and will obscure relationships between queries and documents that involve these terms. When the domain is more technical, one cannot always rely on thesauri to solve this problem. Instead, the distribution

of terms have to be inspected to ensure that terms that are similarly distributed are treated similarly in document retrieval.

Likewise, sometimes a particular term/dimension inappropriately conflates semantically different concepts via *polysemy*. An example of this is, again, "chair"—which refers both to a seat and to the leader of a committee.

One technique that is currently used to both reduce and separate dimensions is *Latent Semantic Indexing* (LSI). LSI uses a linear algebra technique known as *singular-value decomposition* (SVD) to map the dimensions of the original document vector space into dimensions called "concepts", which represent associations of terms. It is a technique that projects document term vectors into a space with "latent" semantic dimensions, where co-occuring terms are projected into the same dimensions and non-co-occurring terms into different dimensions.

Let M be a matrix whose n rows represent terms and whose m columns represent documents. In effect, the contents of each column is a document vector. Then the SVD of M is represented by

$$M = USV^T$$

where U is a term-by-concept (n by m) matrix, V^T is a concept-by-document (m by n) matrix, and S is a square (m by m) matrix containing, along the diagonal, the singular values that represent the relationships between terms and documents. U and V are both orthonormal. The values in S descend diagonally by magnitude. Intuitively, SVD converts the direct relationship between terms and documents into an indirect one mediated by concept associations and scaled by singular values, which reflect the relative importance of the concept associations.

The columns of V^T can be used as substitute document vectors for similarity computation where the dimensions are concepts. They correspond to the original document vectors transformed to the new concept space.

Since many of the concept associations are trivial, they create noise. V is orthonormal, and each concept's contribution to a document vector is unweighted by importance. The corresponding S matrix, however, contains ranked coefficients that, intuitively speaking, determine the importance of the concepts in descending order. Consequently, one can establish a cutoff in the ranks and reduce the number of rows in V^T accordingly to only consider the significant concepts, before performing similarity computations between contracts and templates in that matrix. The cutoff point (the number of top concepts to consider) has to be determined empirically.

3.3 Use of Lucene

In our system Lucene is used only as a convenient and efficient means of converting documents into term vectors and for extracting statistics about the corpus for computing $tfidf$. Lucene itself cannot be reliably used to compare contracts with templates using simple cosine similarity since it does not report similarity scores using a strict cosine measure from 0 to 1, and there is no guarantee that Lucene's scores will hold stable across different sets of documents.

3.4 Term Vector Creation

To create term vector representations of each document, the contracts and template files are first used to create two separate Lucene indices. Next, the terms present in the template index are extracted using Lucene APIs, and these form the term space for converting template and contract documents into term vectors.

The use of only template terms for term vector creation is a simple approach for overcoming a key class of OCR errors - the presence of non-English (misspelt) words introduced by the image–to–text conversion process. By limiting the term vector terms to only those present in templates (which are cleanly formatted and spell-checked Microsoft Word documents), misspelt terms in contracts are automatically ignored. Note that this is not the best possible solution; a better alternative would be to attempt to clean such terms via spelling correction techniques so that the presence of the term in the contract is not ignored completely. However, the above filtering approach does produce better results than considering terms in templates as well as contracts without spelling correction.

A number of alternatives were explored for computing the term weights - beyond using the standard $tfidf$ formula. The first was *IDF smoothing*, and the other was *term boosting*.

3.4.1 IDF Smoothing

In typical search systems, the IDF component of the term weight ensures that a relatively rare term, which occurs only in a few documents, results in those documents being ranked most prominently in the search result when a query contains that term. However, this property of IDF has an adverse effect when the vector-space model is used to compare two documents. Consider a term such as "health". It was observed that the occurrence of that term in the name of the customer in a given contract (a relatively rare event) made that document appear to be highly similar to a template whose subject was related to "network health checking", which was a false conclusion. While the bias introduced by high IDF values for rare terms helps in search to retrieve relevant documents, it seems to have an adverse effect when comparing two documents. To counteract this effect, a smoothing technique was used to reduce IDF values that were too high. The formula used for smoothing is:

$$idf(i) = \log \frac{|D| * (1 + C)}{|\{d : d \in D \ \& \ t_i \in d\}| + |D| * C}$$

where D is the set of documents and C is a constant that was set to 0.1 and 0.2 for different experiments (see [8] for the intuition behind this formula).

3.4.2 Term Boosting

For term boosting, in each document processed, certain domain-specific terms were identified as being more important than others, and their weights were increased according to the formula below:

$$\max\left(tf(i) * 2, maxtf\right)$$

Essentially, the weight of a domain-specific term is doubled, but capped by the value of the highest term weight in the document before boosting. In the experiments conducted, the domain-specific terms chosen for boosting were those that appeared in the document titles and section headings of all contract templates that were part of the corpus used for the experiments. Examples terms include General Responsibilities, Transition, Order Management, and so on.

3.5 Latent Semantic Indexing

For LSI experiments, NIST's public domain Java linear algebra package named JAMA was used for dimensionality reduction. JAMA contains a built-in SVD implementation. Code for computing cosine similarity of two document vectors, over the LSI-reduced concept space as well as the original term space, was implemented as part of the system.

4. EXPERIMENTS AND RESULTS

4.1 Evaluation Criteria

In this section, we compare the performance of the LSI-based and cosine similarity-based approaches. The data consists of a set of 340 contracts and 50 templates, for the domain of communication services. The SVD matrix has about 3240 terms. We manually identified the templates matching each contract (or the fact that there was no matching template) as the ground truth and compared it to the results from both LSI and cosine similarity.

When identifying the ground truth data set, we observed that most of the contracts have at most one matching template. Additionally, since the next component of the system performs further analysis on the selected templates to compute the deviation between the contract and each template, false positives introduced by the content similarity-based filter will eventually be filtered out. Therefore, only recall is used to measure performance without penalizing false positives. Here, recall is defined as the number of returned matching templates divided by the total number of matching templates.

4.2 Evaluation of Results

For each contract, we compute its similarity score with each of the templates using both LSI and cosine similarity techniques, and rank the templates based on the scores. Then we retrieve the list of Top-K templates and the list of templates whose similarity scores are higher than a given threshold. In Table 1, we compare the recall of LSI with that of cosine similarity for the Top-K template retrieval, where K was 1, 3 and 5 respectively. The comparison with regard to the threshold based template retrieval is shown in Table 2, where we did experiments for thresholds 0.9, 0.75 and 0.6.

In both tables, the columns with heading "None" represent the measurement of recall without applying any special processing technique to the term weights. Both LSI and cosine similarity do not perform very well in that case, expecially for the threshold based approach. In order to improve the matching quality, we used IDF smoothing with a factor of 0.2 and boosted weights of domain specific terms in document titles before computing similarity scores. The columns with heading "IDF" present the results of using IDF smoothing, the columns with heading "Boost" show the results of boosting term weights and the columns with heading "IDF & Boost" document results when both IDF smoothing and term boosting are used.

In both retrieval approaches, IDF smoothing and term boosting have improved the performance of LSI and cosine similiary. Especially for LSI, when the two processing techniques are used together, it reaches the highest recall in all of the cases. On the other hand, in the Top-K approach, whenever K is increased, the recall is increased (because more templates are being retrieved). Similarly, in the threshold based approach, whenever the threshold is descreased and more templates are retrieved, the recall is increased.

Compared to LSI, cosine similarity returns relatively low similarity scores for the templates as shown in Table 2, but it can quickly identify the matching templates and rank them high as shown in Table 1. Therefore, LSI appears to be better when used in a threshold-based filter while cosine similarity is better for the Top-K approach. However, it should be noted that when the threshold is set to be low, the LSI approach may retrieve many templates for a given contract instance, which can slow down the subsequent Tree Matching component. We need to balance the tradeoff between recall performance and the the number of templates selected for detailed structural analysis.

4.3 Boosting LSI Scores

Another experiment we did was to combine LSI based and cosine similarity based techniques to further improve recall. The idea is to boost LSI scores based on cosine similarity results. If cosine similarity ranks a template the highest in the results, we boost its corresponding LSI score by using the following formula:

Boosted LSI Score = 0.5 * (LSI Score + 1)

Otherwise, we do not change LSI scores. Table 3 presents the comparison of the recall between LSI and LSI combined with cosine similarity by using the methods of IDF smoothing, term boosting as well as IDF smoothing & term boosting. The threshold was set to 0.75. The performance comparison shows that LSI boosted with cosine similarity consistently outperforms LSI for all of the three methods.

4.4 Observations

For some contracts, LSI ranked non-matching templates higher than matching templates. One of the reasons could be that only a few hundred documents were used to perform SVD while typically, thousands of documents are used to enable LSI to accurately detect co-occurring terms. With a small number of documents, LSI may mistakely group irrelevant words together to form concepts.

In addition, sometimes, both LSI and cosine similarity identified incorrect templates due to the complexity of the compliance problem. In some cases, a contract and its best matching template comply perfectly at a structural level (i.e., they have very similar section headings), but the contract has section content that differs from that of the matching section in the template, possibly due to tailoring of the template for specific client requirements. In other cases, a contract does not include all of the sections in the nearest matching template, or adds a few sections that the best matching template does not have. Such customization confuses both LSI and cosine similarity since they cannot take document structure into account. We believe that boosting weights of domain specific terms in document titles helps reinforce some degree of structural information. Therefore, that technique has led to the improvement in performance.

5. RELATED WORK

The problem of matching documents by their textual content to templates, in order to assess compliance with regimes specified in those templates, is not one that is well-studied in the literature. There has been considerable work in the

	LSI				Cosine Similarity			
	None	**IDF**	**Boost**	**IDF & Boost**	**None**	**IDF**	**Boost**	**IDF & Boost**
Top-1 recall	0.3	0.5	0.6	0.6	0.6	0.8	0.8	0.7
Top-3 recall	0.7	0.8	0.8	0.8	0.8	1	1	1
Top-5 recall	0.9	0.9	1	1	0.8	1	1	1

Table 1: Recall for LSI versus Cosine Similarity w.r.t. Top-K Template Retrieval

	LSI				Cosine Similarity			
	None	**IDF**	**Boost**	**IDF & Boost**	**None**	**IDF**	**Boost**	**IDF & Boost**
Threshold-0.9 recall	0.3	0.4	0.3	0.4	0.1	0.1	0.1	0.1
Threshold-0.75 recall	0.3	0.5	0.5	0.6	0.3	0.4	0.3	0.5
Threshold-0.6 recall	0.6	0.7	0.7	1	0.4	0.5	0.5	0.5

Table 2: Recall for LSI versus Cosine Similarity w.r.t. Threshold based Template Retrieval

space of image pattern recognition as in Peng et al. [9]. They present an algorithm that uses Component Block Projections (CBP) of document images to find the correct template image out of a large set of templates. Other work using image-based techniques include Hu et al. [10] and Shimotsuji and Asano [11]; the latter use the locations of cells on form images to match them to each other. Our work involves templates that are already in text format, and text documents whose quality after OCR processing is highly variable.

Perhaps more closely related is the work of Minakov et al. [12] who developed a multiagent system for text analysis that clustered auto insurance contracts by semantic features. Features extracted from the clusters were then used to assist domain experts in constructing insurance contract templates. However, in our work, the templates are already created by domain experts, and we use IR techniques to find the best matching template given service contracts.

Brauer et al. [13] attempt to match data from unstructured business documents to structured enterprise data stored in databases. However, while contract templates have some structure, they are not structured data in a relational database for example.

Chen et al. [14] use "template matching" to retrieve patent documents based on queries, but these "templates" are actually a means of representing the syntax and semantics of expressions in patent abstracts. They are not classifying entire documents by the characteristics of whole template documents.

6. CONCLUSIONS AND FUTURE WORK

In any effort to convert IT service offerings into commodities, standardization is a key ingredient. The main contribution of this paper has been to demonstrate the use of a novel paradigm in a system developed for checking the degree of contract standardization: namely, the use of information retrieval techniques to efficiently identify the degree of compliance of service contracts with a set of templates. In the pursuit of this goal, we tested two information retrieval techniques, using a paradigm of query-as-classification. The classification task was to efficiently determine, given a contract document, on which template it had been based. However, given the nature of service contract customizations and what constitutes compliance (a strong focus on structural match), it is often the case that a purely document level similarity analysis is insufficient for identifying the compliance to a given template. Thus, this work presents the preliminary step in classifying contract documents by template: candidate discovery.

We selected and tested some variants of cosine similarity and LSI as our information retrieval technique. There were two ways of doing this: selecting candidates by similarity rank, and selecting candidates by thresholding similarity scores. We considered success to be when the correct template appears in the rank- or threshold-filtered search results.

The comparison is complex: there are some contexts in which LSI works better than cosine similarity, and there are some in which cosine similarity works better than LSI. If we filter search results by rank (the Top-K approach), then on the whole, cosine similarity is more successful; however, the effect disappears when we increase the number of ranked documents permitted to appear in the search results, in which case the LSI-based and cosine similarity-based approaches achieve parity. The reverse is true if we use a threshold-based filter; in fact, only LSI gets 100% recall in this category.

If we consider the goal of the filtering step to be the delivery of a restricted set of candidates to later phases of the compliance-analysis processing pipeline, then we should prefer the use of Top-K—and therefore the cosine similarity approach. This is what concern over efficiency would dictate.

However, if future efficiency were not a consideration, and we were to consider the value of finding more candidate templates while being able to rule out any template for a completely mismatching contract, then the threshold approach would have an advantage. With a variable number of plausible candidate templates and a sufficiently low threshold, the risk of a false negative is reduced. In spite of such a threshold, if zero contracts make the cut, then no detailed analysis will be necessary whereas in the top-K approach, there are always K templates to consider in the next phase. Then we see that LSI would have an advantage over cosine similarity.

Some of the improvements for both LSI and cosine similarity are related to the introduction of IDF smoothing and term boosting, which suggests that these effects are actually due to better weighting of the terms.

One key problem with whole document comparison is that even for contracts that are associated with a single type of service, there can be natural differences between them. An inventory (parts) list for a service contract would vary from one customer to another, and nonstandard parts of a contract such as the cover letter and signature pages, if present in one contract and not another, will confuse any whole document similarity measure. In the future, we plan to use basic document structure identification techniques to determine

	LSI	LSI + Cosine Similarity Rank
IDF Smoothing	0.5	0.7
Term Boosting	0.5	0.8
IDF Smoothing & Term Boosting	0.6	0.8

Table 3: Recall for LSI versus LSI + Cosine Similarity Rank w.r.t. Threshold-0.75 Recall

if such sections exist, and prune their content before computing a term vector representation of the document. Such techniques (e.g., used in the Segment Analyzer component in Figure 2) are also required for the detailed similarity computation of a contract with the candidate templates. However, an efficient version of the segment analyzer, that only focuses on identifying sections prone to introducing noise in whole document similarity computations, has to be developed.

7. REFERENCES

[1] Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34** (2002) 1–47

[2] Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In Kontkanen, P., Tirri, H., Silander, T., Myllymäki, P., Zheng, Z., eds.: Machine Learning: ECML-98. Lecture Notes in Computer Science, Chemnitz, Germany, Springer (1998) 137–142

[3] McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification (1998)

[4] Mayoraz, E., Alpaydin, E.: Support vector machines for multi-class classification. In: Proceedings of the International Workshop on Artifical Neural Networks (IWANN99), IDIAP, Springer-Verlag (1999) 833–842

[5] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science **41** (1990) 391–407

[6] Reeves, D.M., Wellman, M.P., Grosof, B.N.: Automated negotiation from declarative contract descriptions. In Müller, J.P., Andre, E., Sen, S., Frasson, C., eds.: Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, ACM Press (2001) 51–58

[7] Bagby, J., Mullen, T.: Legal ontology of contract formation: Application to eCommerce. In: "AAAI Workshop on Contexts and Ontologies", Pittsburgh, PA, USA (2005)

[8] Robertson, S.: Understanding inverse document frequency: On theoretical arguments for idf. Journal of Documentation **60** (2004) 503–520

[9] Peng, H., Long, F., Chi, Z.: Document image recognition based on template matching of component block projections. IEEE Trans. Pattern Anal. Mach. Intell. **25** (2003) 1188–1192

[10] Hu, J., Kashi, R., Wilfong, G.: Document image layout comparison and classification. In: ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition, Washington, DC, USA, IEEE Computer Society (1999) 285

[11] Shimotsuji, S., Asano, M.: Form identification based on cell structure. In: ICPR '96: Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume III-Volume 7276, Washington, DC, USA, IEEE Computer Society (1996) 793

[12] Minakov, I., Rzevski, G., Skobelev, P., Volman, S.: Creating contract templates for car insurance using multi-agent based text understanding and clustering. In: HoloMAS '07: Proceedings of the 3rd international conference on Industrial Applications of Holonic and Multi-Agent Systems, Berlin, Heidelberg, Springer-Verlag (2007) 361–370

[13] Brauer, F., Löser, A., Do, H.H.: Mapping enterprise entities to text segments. In: PIKM '08: Proceeding of the 2nd PhD workshop on Information and knowledge management, New York, NY, USA, ACM (2008) 85–88

[14] Chen, L., Tokuda, N., Adachi, H.: A patent document retrieval system addressing both semantic and syntactic properties. In: Proceedings of the ACL-2003 workshop on Patent corpus processing, Morristown, NJ, USA, Association for Computational Linguistics (2003) 1–6