

On Extracting Structured Knowledge from Unstructured Business Documents

Gaurav Pandey

Department of Computer Science
University of Minnesota, Twin Cities
Minneapolis, MN, USA
gaurav@cs.umn.edu

Rakshit Daga

Design Services Team
SAP Labs LLC
Palo Alto, CA, USA
rakshit.daga@sap.com

Abstract

Efficient management of text data is a major concern of business organizations. In this direction, we propose a novel approach to extract structured knowledge from large corpora of unstructured business documents. This knowledge is represented in the form of object instances, which are common ways of organizing the available information about entities, and are modeled here using document templates. The approach itself is based on the observation that a significant fraction of these documents are created using the *cut-copy-paste* method, and thus, it is important to factor this observation into business document analysis projects. Correspondingly, our approach solves the problem of object instance extraction in two steps, namely similarity search and then extraction of object instances from the selected documents. Early qualitative results on a couple of carefully selected document corpora indicate the effective applicability of the approach for solving an important component of the efficient text management problem.

1 Introduction

Text is probably the most common form for knowledge in today's world. All forms of useful information in various domains, be it education, business or education, gets published as books, webpages, papers or some other form of text. Especially, in the domain of business, documents are an integral part of the process, since every official detail has to be documented for various purposes, such as sharing and dissemination, keeping proofs of decisions made and standardizing processes. Thus, it is very important for organizations working in this domain to manage the information contained in these documents effectively.

However, despite this importance, an inherent problem with text data is that, in many cases, it is unstructured, i.e., there is no standard format in which information is recorded in a document. This makes it extremely hard for a computer to automatically extract useful knowledge from a document, and thus, business organizations have to employ large amounts of specialized human labor for this task. More so, in large organizations, where the volume of text data is usually

very large, even this labor proves to be insufficient. Hence, there is a great need for an automated system that can extract structured knowledge from unstructured text documents. This paper presents a text mining approach that achieves this target for a variety of documents.

One useful method to model the structure in a document is via objects. An object is an entity that can be described in terms of individual attributes. For instance, *person* is an object whose attributes can be *name*, *nationality*, *profession* and *income*. Instances of this object are persons who have a specified value for each of these attributes. Clearly, once these object instances have been specified, the information so gathered can be stored persistently in an easily accessible and retrievable structure, such as a relational database.

Corresponding to this utility of the notion of an object, we hypothesized that many business documents describe instances of different types of objects, such as stock purchases, personnel records, business contracts and others. Thus, the solution proposed in this paper attempts to extract these object instances from documents, so that they can be stored in appropriate storage structures, such as databases, and accessed easily. This task is achieved by modeling an object using a document template and breaking the overall solution into two steps:

1. **Similarity search** to identify which documents in the given corpus have been created from this template.
2. **Extraction** of instances from the identified documents.

Another useful insight that assisted the implementation of these steps was the method of creation of the documents, i.e. *direct vs indirect* creation. Briefly, the former method refers to the *cut-copy-paste* method of document creation, while the latter indicates the use of the template as a guide for the preparation of the document. In accordance with this insight, the traditional document similarity is modified to include a *diff*-based similarity, which estimates the extent of direct creation in a document. This similarity is also used in the extraction step to identify the most likely attributes values for object instances in a document. The complete approach is described in detail later.

The rest of the paper is organized as follows. Section 2 discusses related work in the field of structure extraction from text. Section 4 discusses our approach for this problem in detail, and Section 3 explains the necessary background con-

cepts. Some early qualitative results on two document corpora are presented in Section 5. We conclude with directions of future work in Section 6.

2 Related Work

The field of information extraction has been around for a long time [Pazienza, 1999], though most of the focus in this area was initially on using natural language processing-based techniques for this task [Strzalkowski, 1999]. One of the earliest works in this direction was the recognition of named entities in text [Borthwick, 1999; Cohen and Sarawagi, 2004]. These approaches used supervised techniques such as HMM and maximum entropy classification for identifying proper nouns and their corresponding entities in text. Since supervised techniques become infeasible for large data sets because of their requirement of additional knowledge about the data, we have adopted an unsupervised approach in this work. This enables us to accomplish the task in a larger scale, such as text repositories of a large business organization.

The closest work to ours has been presented in the database community [Mansuri and Sarawagi, 2006; Borkar *et al.*, 2001; Chakaravarthy *et al.*, 2006]. While [Borkar *et al.*, 2001] modified the hidden Markov models to find the best segmentation of a given text document into structured records, [Mansuri and Sarawagi, 2006] used conditional random fields to identify various recognition clues in the data, and integrating knowledge about entities available in relational databases. [Chakaravarthy *et al.*, 2006] also use knowledge about entities available in databases, and aids this with entity templates to create links between the database and the given unstructured text document. Thus, even though these approaches are very close in spirit, our approach is more widely applicable, since it is unsupervised and does not assume any external knowledge in the form of databases.

In addition to the above research in the field, work has also been done in relation to the constituents steps of the overall approach. We discuss these works during the discussion of the corresponding step.

3 Background

3.1 Latent Semantic Indexing (LSI)

A very crucial task for computational text analysis is representing a document in the Euclidean space so that important operations, such as similarity computation between documents, can be performed conveniently. Latent Semantic Indexing (LSI) [Deerwester *et al.*, 1990] has emerged as the most popular technique for this task, since it very effectively handles the natural language problems of synonymy and polysemy, besides providing mathematical benefits such as a noise-resistant vector representation of documents. In brief, LSI involves the singular value decomposition of the original term-document matrix $X_{t \times d}$ to give three new matrices U , Σ and V according to equation 1. Here, Σ is a diagonal matrix containing the m eigen values of X , while U and V contain the left and right eigen vectors of X .

$$X_{t \times d} = U_{t \times m} \Sigma_{m \times m} V_{m \times d}^T \quad (1)$$

Now, if only the k biggest eigen values of X are retained in Σ , and U and V are modified accordingly, then the refined term-document matrix \hat{X} is obtained as in equation 2 is shown to contain much less noise and more informative vector representations for the documents than X [Deerwester *et al.*, 1990; Berry *et al.*, 1994].

$$\hat{X}_{t \times d} = \hat{U}_{t \times k} \hat{\Sigma}_{k \times k} \hat{V}_{k \times d}^T \quad (2)$$

Finally, given the above transformation of X , the term vector of any text query can be transformed as per equation 3.

$$\hat{X}_{query} = X_{query}^T \hat{U} \hat{\Sigma}^{-1} \quad (3)$$

Once this transformation is complete, the similarity computation of this query with any document in the corpus can be performed just by carrying out a dot product of this query vector with the vector in \hat{X} corresponding to the document. LSI has to been shown to be very accurate at this task [Berry *et al.*, 1994], and is the most heavily used application of LSI in our work.

3.2 Diff Algorithm

A very early problem in text analysis was the comparison of the content of two documents at the syntactic level, i.e., whether the same terms have been used in the document, and if so, how do the sentence composed by them differ. The most efficient solution to this problem was the Diff algorithm [Ukkonen, 1985], which was very similar to the string edit distance algorithm [Needleman and Wunsch, 1970]. The algorithm uses dynamic programming to find the maximal pieces of text that are identical between two documents, and categorized the mismatched portions as *insertions*, *deletions* and *replacements*. Thus, the output of the algorithm describes the syntactic differences between two documents, and thus is an integral part of our solution. Details and pseudo-code of Diff are omitted here for brevity purposes, but can be found elsewhere [Hunt and McIlroy, 1976]. Also, Diff is very commonly used as the *diff* command in UNIX systems.

3.3 Inverse Document Frequency (*idf*)

Often in a large document corpus, there exist a significant number of terms that occur very rarely, and thus are very important for discriminating the documents they appear in. Several term weighting schemes have been proposed to model this behavior, of which, the inverse document frequency (*idf*) [Robertson, 2004] is the most widely used. The *idf* of a term t is defined as in equation 4, where N is the size of the corpus, and N_t is the number of documents containing t .

$$idf_t = \log\left(\frac{N}{N_t}\right) \quad (4)$$

It is clear from the above equation that rare terms, such as proper nouns, are expected to have high *idf* values, while common terms, such as *set*, *large* and *popular* are expected to have significantly low *idf* values. For this accurate modeling of term importance, *idf* has been shown to work well in a variety of text analysis applications, such as classification [Joachims, 1998] and extraction [Salton and Buckley, 1988]. In fact, significant research has been devoted to finding a theoretical justification of *idf* and factors leading to its success [Robertson, 2004].

3.4 Objects and Templates

An object in the business domain is used to capture the available information about an entity, in a structured format [Eeles and Sims, 1998]. A common mathematical view of an object is as a set of attributes. For instance, an *employee* object may be described by the set $\{name, telephone, empnum, position\}$. When specific values are assigned to these attributes, an instance of this object is created. This is the view adopted in our approach, since it gives us an effective way to model the latent structure in a document, i.e., the object instance described in it.

However, the concept of an object is easier formalized than implemented. Given an unstructured document, there are two levels of complexity in identifying the object instance it describes, if there is no external information:

1. It is hard to identify if a term in the document contributes to the description of an object instance.
2. It is hard to find which attribute the term is a value of.

Thus, in order to identify object instances, it is important to model an object more concretely. We adopt the idea of a *document template* for this purpose. A template is a document that specifies, either directly or indirectly, how the attribute values of an object instance are to be incorporated in a document. For instance, an employee file can be used as a template for the *employee* object discussed above. This model is heavily made use of in our approach, as will be seen later.

4 Proposed Approach

With the required background, we now proceed to the details of the proposed approach for solving the problem of extracting structured information from a corpus of unstructured business documents. As mentioned earlier, the target of this approach was the extraction of object instances from a corpus of text documents, where an object is represented by a document template. Given this definition of the problem, our approach solves it in two steps:

1. **Similarity search:** In this step, a similarity search is conducted on the given corpus with the template as the seed document, and the most similar documents are stored in a family. This family of documents are hypothesized to have been created from the template, since they are similar in syntax and semantics to it, and thus expected to contain instances of the given object.
2. **Extraction of attribute values of object instances:** Once the above family of documents has been constructed, the terms in each document are weighted, and potential attribute values of the corresponding object instance(s) are produced.

This approach is illustrated in detail in Figure 4. However, before explaining the above two steps, and their constituent tasks in detail, it is very important to understand two modes of document creation that the corresponding solutions to the steps are based on.

4.1 Methods of Document Creation

It can be observed from experience that a substantial fraction of the documents created by anybody are either directly created from, or with the help of a document that is similar in syntax and/or content. This is especially true in the business world, where various documents have to be prepared regularly, such as contracts, meeting minutes, stock purchases and user interviews and thus are usually prepared on the basis of template(s). Thus, in order to any automated analysis of these documents effectively, these methods of creation have to be incorporated in the analysis procedure. For the purpose of our approach we define two such methods:

1. **Direct creation:** This method models the *cut-copy-paste* mechanism of document creation. In this method, the overall syntax of the document remains the same as the original template, while certain terms are replaced with other specific ones. For instance, a phrase such as *X bought Y shares of Z* in the template may be modified to *Microsoft bought 500 shares of Infosys* using this method. Examples of documents that are created using this method are stock purchases and business contracts.
2. **Indirect creation:** This method involves the use of the semantic content of the original template only as a guideline for the final document. Thus, although the content of the final document is semantically similar to the original document, the syntax may be significantly different. An example in case is that of user research interviews, which are usually prepared on the basis of an interview guideline that provides the questions to be asked to the user and observations to be made.

This dichotomization of the document creation process is a key insight utilized by our approach and is thus incorporated carefully into the solutions of both the similarity search and the instance extraction steps. This is achieved by introducing a Diff-based similarity measure into the similarity estimation process and use of this similarity to identify the best field values for an object instance. Details of how this is modeled mathematically are presented in the new few sections.

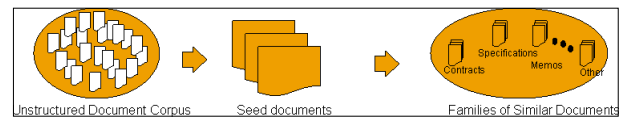


Figure 1: Overview of the similarity search step

4.2 Similarity Search

The first step of our approach is the identification of documents in the corpus that have been created from the template representing the object under consideration, as shown in Figure 1. The hypothesis underlying this step is that documents that are *similar* to the template have been created from the template, and thus, are highly likely to contain instances of the corresponding object. In order to model this hypothesis accurately, it is necessary to use a similarity measure that reflects the above methods of document creation from the template. For this purpose, we use multiple measures to estimate

different types of similarities between two documents, and derive an overall similarity estimate from these values. Following is a detailed description of each of these measures, how they are combined, and how the final set of most similar documents is selected. However, before discussing these methods, it is important to note that one of other possible approaches for building the families of documents similar to the given templates is clustering [Jain and Dubes, 1988]. We do not follow this approach for the following two reasons:

1. One document can be in multiple families, since its preparation may have involved multiple templates.
2. All similarities are calculated with respect to a template, and not between documents, as is done in clustering.

With this clarification, we proceed to the discussion of the similarity measures used in our approach.

LSI-based similarity

This similarity measure was designed to evaluate the semantic similarity of two document, since this indicates if one of the documents may have been *indirectly* created from the other, as defined above. Since LSI is a powerful technique to quantify the semantic content of a document, an LSI-based similarity measure named sim_{lsi} is used for this task. This measure simply computes the dot product between LSI vectors α_1 and α_2 of two documents d_1 and d_2 respectively, as shown in equation 5.

$$sim_{lsi}(d_1, d_2) = \alpha_1 \cdot \alpha_2 \quad (5)$$

In our implementation, LSI was applied to the original *tf – idf* matrix for the corpus [Salton and Buckley, 1988], where true frequency (*tf*) takes into account the occurrence, and inverse document frequency (*idf*) accounts for the importance of each term in a document. Also, the implementation of LSI and the similarity calculation were adopted from the Generic Text Parser (GTP) [Giles *et al.*, 2003] software, and the default parameters therein were used for these tasks.

Diff-based similarity

sim_{lsi} indicates if one document was indirectly created from another. However, if a similar evaluation has to be done for the *direct* creation method, then, as per the definition, the syntactic similarity of the documents has to be evaluated, i.e., how equivalent are the composition of the sentences, and the order of the words in them.

The solution adopted for this problem is derived from the Diff algorithm discussed earlier. Assume that documents d_1 and d_2 have been processed by the Diff algorithm. Now, as per the output of this processing, let S_1 be the set of terms in d_1 that have been either replaced or deleted, and let S_2 be those in d_2 that have been inserted, then the diff-based similarity $sim_{diff}(d_1, d_2)$ can be calculated as in equation 6.

$$sim_{diff}(d_1, d_2) = \min\left(\frac{|d_1| - |S_1|}{|d_1|}, \frac{|d_2| - |S_2|}{d_2}\right) \quad (6)$$

This definition of sim_{diff} essentially implies that it is inversely proportional to the maximum fraction of change between the two documents. Thus, this measure indeed measures how much of the syntax of a document has been modified to transform it into the other document. The idea was

implemented using a modified version of the Diff program of [Darwin and Lindsay, 2006].

Tag-based similarity

For many documents that have been prepared by human experts, such as user research interviews and pictures, tags are often assigned to ensure efficient organization in a repository. Since many such documents are expected to be encountered in the business domain, and to make use of human intelligence wherever available, we also incorporated a tag-based similarity into the overall measure. If T_1 and T_2 are the sets of tags for documents d_1 and d_2 respectively, then the tag-based similarity sim_{tag} , between d_1 and d_2 can be defined as in equation 7b.

$$T_{common} = \{t | t \in T_1, t \in T_2\} \quad (7a)$$

$$sim_{tag}(d_1, d_2) = \min\left(\frac{|T_{common}|}{|T_1|}, \frac{|T_{common}|}{|T_2|}\right) \quad (7b)$$

Thus, sim_{tag} counts the fraction of tags that are the same between two documents, and assigns the similarity as the minimum of the two fractions. This ensures that the calculation is not biased towards documents that may have fewer tags than others.

Overall similarity

The previous sections discussed the computation of three different similarity measures between a pair of documents, namely LSI-based similarity sim_{lsi} , Diff-based similarity sim_{diff} and tag-based similarity sim_{tag} . However, once these similarities have been calculated, the need arises to combine them to calculate an overall similarity. Our approach to this problem uses a simple weighted mean of the similarities. This operation is mathematically valid, since the range of each of these similarities is $[0, 1]$, and thus, the overall similarity also has the same range. The precise weights are pre-specified using the idea that both the direct and indirect methods are expected to be of the same importance for document creation, and thus their corresponding similarities, sim_{diff} and sim_{lsi} respectively, are assigned equal weights. Also, wherever available, tag-based similarity sim_{tag} is also assigned the same weight as the others, so as to give importance to human intelligence. Using these ideas, the overall similarity sim is computed as in equation 8. The two cases in this equation account for the conservative assumption made that $sim_{tag}(d_1, d_2) = 0$ implies that no tags were assigned for d_1 and d_2 , and thus, this similarity should not be considered in the weighted mean calculation. Since tags are derived from a document's content and syntax, the other similarities are expected to account for any exceptions to this assumption.

$$sim(d_1, d_2) = \begin{cases} \frac{\sum_{i \in \{diff, lsi\}} sim_i(d_1, d_2)}{2} & sim_{tag}(d_1, d_2) = 0 \\ \frac{\sum_{i \in \{diff, lsi, tag\}} sim_i(d_1, d_2)}{3} & \text{otherwise} \end{cases} \quad (8)$$

Thus, once this similarity computation is carried out with respect to a template for all the documents in a corpus, the latter can be arranged in the order of decreasing similarity values. The question that arises now is how to select the most

similar documents out of this listing. The approach used to solve this problem is discussed in the next section.

Selection of Most Similar Documents

Our approach to solve this problem is based on the observation that the distribution of similarity values in a corpus of business documents is skewed, i.e., most of the documents are expected to have a small similarity with the template, while a small number of documents that have been created from the template will have significantly higher similarities. This observation is validated by the distribution shown in Figure 2 that is generated for a corpus of user research interviews, and this is the trend observed for several other corpora.

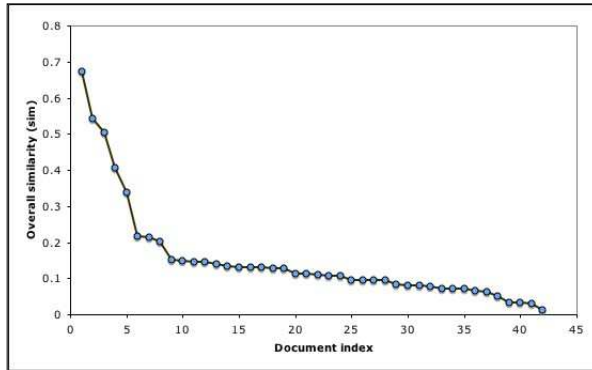


Figure 2: Distribution of *sim* values for a sample corpus and template

Clearly, under such a distribution, it does not make sense to select a fixed number of documents. Thus, in keeping with the observation, the set of most similar documents, say S , are selected on the basis of the area under the distribution curve. Starting with the document with the highest similarity, documents are added to S , until the sum of their corresponding similarities is half that of the similarities of all the documents in the corpus. Thus, this method adaptively selects the set of most similar documents, i.e. S , for a given template.

The previous subsections detailed how, given a template corresponding to an object definition, a set of most documents have been created using it, and thus are expected to be syntactically and/or semantically similar to it, can be retrieved from a corpus. Also, since a corpus is expected to contain documents describing instances of multiple objects, the modularity of the similarity search step allows the specification of multiple templates and the retrieval of their corresponding sets of similar documents. This enables the realization of the scenario shown in Figure 1, where multiple templates, or seed documents, are used to build corresponding families of similar documents.

4.3 Extraction of Attribute Values of Object Instances

Once a family of documents that have been created from a given template, and are thus each expected to contain at least one instance of the object, has been constructed, the next task

of our approach is the retrieval of these object instances from the constituent documents. Since it is hard to train a computer to identify individual attributes of an object because of reasons noted earlier, and all possible values for those attributes, we took an unsupervised approach to this task. This approach consists of the measurement of significance of the terms for a given document, and the display of the most significant terms to the user. Thus, given the object definition, it is hoped that the user will construct the object instance by mapping the terms produced with the attributes of the objects. However, our approach ensures that suggestions made are robust and not very large in number, and the first step in this direction is the use of the inverse document frequency (*idf*) scoring scheme for terms.

As mentioned before, *idf* is an effective scheme for estimating the significance of a term for the documents it appears in [Salton and Buckley, 1988; Robertson, 2004]. This is so, since a term that occurs in a small number of documents, and thus is very significant for differentiating them, will have a high *idf* as per equation 4, and vice versa. Due to this strong reasoning, we also use *idf* for evaluating the significance of a term, and sort all the terms in a document in decreasing order of their *idf* values. However, a set of significant terms has to be selected from this list to be shown to the user. This selection is done in keeping with the method that is expected to have been used for creating this document. The following sections discuss how these *direct* and *indirect* methods are incorporated into the object instance retrieval process.

Direct Case

If a document d has been created by the method of direct creation from a template t , this implies that some terms that were inserted into d by replacing terms in t . This is important, since these inserted terms are expected to be specific to d , and thus are likely to be the attribute values of the object instance being described in d . For instance, in the example discussed earlier, namely the transformation from *X bought Y shares of Z* to *Microsoft bought 500 shares of Infosys*, *Microsoft*, *500* and *Infosys* may be the values of the *buyer*, *quantity* and *seller* attributes of an instance of the *stock_purchase* object. Thus, this method of document creation was handled by treating these terms as potential attribute values.

The above idea was implemented as follows. Firstly, the document d was assigned to this case if $sim_{diff}(d, t) \geq \gamma$, where $\gamma \approx 1$, which indicates that d was created from t by replacing a few terms. Next, all the terms that were inserted into t , as output by running the Diff algorithm on d with respect to t , are ranked in terms of their *idf* values in descending order. Since $sim_{diff}(d, t)$ is very high, we expect this list to be small, and thus, the results will provide robust and not too many suggestions for possible attribute values of the object instance described in this document.

Indirect Case

In the indirect case, a document d is expected to have been created using a template t only as a guide. Thus, the problem of object instance extraction becomes harder here. This is so, since in this case, it is not clear which terms are expected to be attribute values. Thus, here, all the inferences have to be made on the basis of the significance of a term, since the

terms that are more specific to this document are expected to be specific details of the object instance being described. This idea is implemented using the following steps.

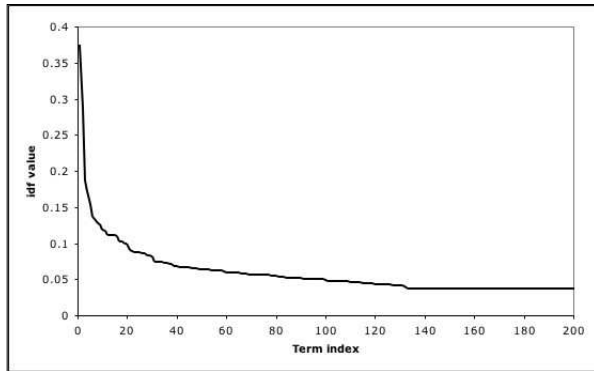


Figure 3: Distribution of idf values of terms in a document

To start with, all the terms in the document are sorted in terms of their idf values. Again, this distribution of idf values is found to be skewed, as shown in Figure 3 for a sample document. This is understandable, since only a few terms in the document are expected to be significant, and most of the other terms are expected to be relatively more common. Thus, a procedure similar to the one used for selecting the most similar document documents to a template is used here. However, since terms in a document are expected to be more numerous than documents in a corpus, and since most of the terms are expected to have very low idf , we selected only the highest ranking terms that contribute a fifth to the total sum of idf values for all the terms in the document. This provides us a list of the most significant terms for the document that are expected to denote attribute values of the object instance described therein.

Grouping of Significant Terms

In the sections above, we discussed how to select the most significant terms in the two cases of direct and indirect creation of a document from a template. However, in many cases, groups of terms may denote attribute values, such as full names like *George Bush*, instead of *George* and *Bush* separately. Thus, combinations of these terms are also considered as potential attribute values, by estimating an idf value for the phrases so formed. Earlier approaches estimated the idf values of phrases by treating constant length phrases as terms [Mitra *et al.*, 1997; Andrews *et al.*, 1998]. However, we perform this estimation with the help of Lemma 4.1.

Lemma 4.1 *If $T = \{t_1, t_2, \dots, t_n\}$, where t_i ($1 \leq i \leq n$) are terms, then $idf(T) \geq \max(idf(t_1), idf(t_2), \dots, idf(t_n))$.*

Proof. Using the notation of equation 4, and noting that the frequency of T will be less than or equal to any of the terms t_i , we have that $idf(T) = \log\left(\frac{N}{N_T}\right) \geq \log\left(\frac{N}{\min(N_{t_1}, \dots, N_{t_n})}\right) \geq \max\left(\log\frac{N}{N_{t_1}}, \dots, \log\frac{N}{N_{t_n}}\right) = \max(idf(t_1), idf(t_2), \dots, idf(t_n))$. Hence proved.

In our approach, in keeping with the above result, $idf(T)$ is set conservatively to its lower bound, i.e., $\max(idf(t_1), idf(t_2), \dots, idf(t_n))$, and the output is reorganized as follows in the two cases:

1. *Direct case:* By definition, the Diff algorithm produces maximal phrases that are different between the two files. Thus, in this case, these phrases are simply ranked using their idf as estimated above, and output to the user.
2. *Indirect case:* Once the set of significant terms has been produced, terms that occur contiguously in the original document, are combined into a phrase, which is in turn ranked according to its idf as estimated above.

Thus, at the end of this term combination step, we have a set of robust suggestions of possible attribute values for the object instance described in each document in the family that was constructed in the similarity search step.

4.4 Overall Approach

The previous two sections described in detail the constituent steps of our overall approach for the extraction of structured object instances from a corpus of unstructured documents, namely similarity search and extraction of object instances. Each of these steps constituted multiple cases and sub-steps, which were solved in keeping with the final goal. Figure 4 illustrates the overall approach for extracting instances of an object from a document corpus, given an object's definition. Conducting this process for all the object definitions and their corresponding templates that have been specified by the user, we can numerous instances of objects described in the entire corpus. These instances can in turn be used to populate individual databases for each object, or combinations thereof, as per the need of the data organization problem. Thus, our approach provides a nearly complete solution to this hard problem and is expected to be of great use for organizations grappling with the problem of management of information in the form of unstructured text documents.

5 Experimental Results

The previous section discussed our approach to solve the problem of extraction of object instances from unstructured documents. Even though the approach is intuitively valid, it is necessary to evaluate its performance on real document datasets. We present details of this evaluation on two such corpora. These correspond to the cases of documents created using only the *direct*, and both *direct* and *indirect* methods respectively.

However, it should be noted that, due to the use of free natural language in the text being analyzed, the ultimate validation methodology for this problem is evaluation by a human expert, since the final output of any approach is can be interpreted subjectively. Thus, most of our evaluation is qualitative. Also, since it is hard to conduct a completely quantitative evaluation of the output, we present early quantitative results wherever possible in the following sections.

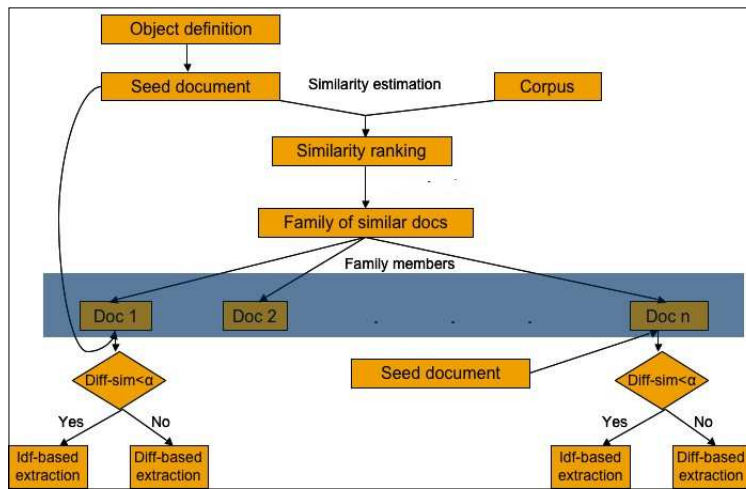


Figure 4: Proposed approach for extracting object instances from unstructured documents (seed document \equiv template)

5.1 The Direct Creation Corpus

It may be noted from Section 4 that a major innovation in our approach is the analysis of documents created using the *direct*, or the *cut-copy-paste* method. Thus, we evaluate our approach separately for this case, using a dataset generated from a cover letter format for visa applications. This letter contains several fields that have to be filled by applicants. We generated a data set of hundred documents by randomly filling these fields with a corresponding set of possible values. We also added hundred unrelated documents to this data set to create a realistic corpus.

Now, for the experiment, an *applicant* object was created consisting of set of attributes $\{name, employer, destination\ country, visa\ type, external\ contact\}$. The original cover letter format was used as the template for this object. Finally, our approach was executed under these settings on the corpus, potential attribute values were extracted from each document, and object instances were composed from these values. Upon a visual inspection of the results obtained, the following observations could be made:

1. All documents created using the template were picked up by the similarity search step.
2. The potential attribute values extracted from a document could be accurately composed into an *applicant* instance. Thus, the precision was high.
3. The coverage was nearly complete, i.e., at least one *applicant* instance was extracted from almost every document.

In further support of the second observation, the following was the observed importance of the various attributes, calculated in terms of the average *idf* value of their values across all documents:

1. *name*
2. *external contact*
3. *destination country*

4. *employer*

5. *visa type*

This order reflected exactly number and frequency of values used to generate the data set. These observations confirmed that our approach handles the direct case very well.

5.2 The General Corpus

In order to show the capabilities of the approach, it was important to evaluate its performance on a general corpus of document, not one prepared by any specific method. For this purpose, we constructed a general corpus by adding forty (proprietary) user research interviews to the corpus created above for testing the direct case. An additional object named *interview* was defined for these documents, its attributes being $\{interviewee, product, problem, solution, helping\ material, keywords\}$. It can be observed that these attributes are not as well defined as those of *applicant*, since the supporting documents were created using a template only as a guideline, and did not follow any particular syntax.

As expected, the results obtained by executing our approach on this general corpus were not as good as those obtained for the previous experiment. However, with sufficient human support, some general trends of good performance could still be observed:

1. Most documents similar to the *interview* template could be retrieved from the corpus, the highest contribution to the similarity coming from *sim_{si}*.
2. Values of most attributes could be retrieved from each document, some attributes having multiple values.

Thus, though this experiment illustrated some new challenges such as the complexity due to the free use of language and multiple values of attributes, the results still showed the potential of our approach for analyzing a general corpus.

We are currently constructing benchmark data sets and evaluation measures that will give us a quantitative evalua-

tion of the results, which will be discussed in a subsequent more detailed paper. Still, based on the above results, we can conclude that our approach holds great promise for solving the problem of extracting structured knowledge in the form of object instances from a corpus of unstructured business documents. This will be a big boost for organizations aiming for efficient management of their text document repositories.

6 Conclusion

In this paper, we presented an effective approach for the extraction of structured knowledge from unstructured documents. The approach was based on the observation that documents are created by two methods, namely *direct* and *indirect* creation. Consequently, it consisted of two steps, i.e., similarity search and extraction of object instances, which took these methods of document creating into account. Early qualitative results on two corpora showed the efficacy of our approach for the stated task.

In future work, we intend to intelligently automate the task of object instance composition from its attribute values, and investigate techniques for incorporating these instances into relational databases. Another important direction would be the design of a quantitative methodology for the evaluation of the performance of the approach. In addition, the problems of estimating the statistical distributions of document similarity and term *idf* values also deserve significant study.

Acknowledgements

The authors thank the SAP Design Services Team for supporting this project. We also thank Himanshu Gupta and the anonymous reviewers for their very constructive feedback on an early draft of this paper.

References

- [Andrews *et al.*, 1998] JE Andrews, TB Patrick, DE Moxley, CM Meyer, M Popescu, and ME Sievert. Using co-occurrence data to determine a thesaurus structure. In *Proc AMIA Symposium*, page 968, 1998.
- [Berry *et al.*, 1994] Michael W. Berry, Susan T. Dumais, and Gavin W. O’Brien. Using linear algebra for intelligent information retrieval. Technical Report UT-CS-94-270, University of Tennessee, Knoxville, TN, USA, 1994.
- [Borkar *et al.*, 2001] Vinayak R. Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. Automatic text segmentation for extracting structured records. In *Proc. ACM SIGMOD*, pages 175–186, 2001.
- [Borthwick, 1999] Andrew Eliot Borthwick. *A maximum entropy approach to named entity recognition*. PhD thesis, 1999. Adviser-Ralph Grishman.
- [Chakaravarthy *et al.*, 2006] Venkatesan T. Chakaravarthy, Himanshu Gupta, Prasan Roy, and Mukesh K. Mohania. Efficiently linking text documents with relevant structured information. In *VLDB*, pages 667–678, 2006.
- [Cohen and Sarawagi, 2004] William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proc. Tenth ACM SIGKDD*, pages 89–98, 2004.
- [Darwin and Lindsay, 2006] Ian Darwin and Donald C. Lindsay. A diff implementation in java. <http://javacook.darwinsys.com/javasrc/textproc/Diff.java>, 2006.
- [Deerwester *et al.*, 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [Eeles and Sims, 1998] Peter Eeles and Oliver Sims. *Building Business Objects*. John Wiley & Sons, 1998.
- [Giles *et al.*, 2003] J. T. Giles, L. Wo, and M. W. Berry. GTP (General Text Parser) Software for Text Mining. In H. Bozdogan, editor, *Statistical Data Mining and Knowledge Discovery*, pages 455–471. 2003.
- [Hunt and McIlroy, 1976] J. W. Hunt and M.D McIlroy. An algorithm for differential file comparison. Technical Report CSTR 41, AT&T Bell Laboratories, 1976.
- [Jain and Dubes, 1988] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.
- [Joachims, 1998] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML ’98: Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, 1998.
- [Mansuri and Sarawagi, 2006] Imran Mansuri and Sunita Sarawagi. A system for integrating unstructured data into relational databases. In *Proc. of the 22nd IEEE Int’l Conference on Data Engineering (ICDE)*, page 29, 2006.
- [Mitra *et al.*, 1997] Mandar Mitra, Christopher Buckley, Amit Singhal, and Claire Cardie. An analysis of statistical and syntactic phrases. In *Proceedings of RIAO-97, 5th International Conference “Recherche d’Information Assistee par Ordinateur”*, pages 200–214, 1997.
- [Needleman and Wunsch, 1970] S.B. Needleman and C.S Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Molecular Biol.*, 48:443–453, 1970.
- [Pazienza, 1999] Maria Teresa Pazienza, editor. *Information Extraction: towards scalable, adaptable systems*. Springer, 1999.
- [Robertson, 2004] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments. *Journal of Documentation*, 60(5):503–520, 2004.
- [Salton and Buckley, 1988] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.
- [Strzalkowski, 1999] Tomek Strzalkowski, editor. *Natural Language Information Retrieval*. Springer, 1999.
- [Ukkonen, 1985] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, 1985.