IJCTA, 9(10), 2016, pp. 4537-4544

A Template Based Checking and Automated Tagging Algorithm for Project Documents

***MG Thushara **Nikhil Dominic**

Abstract : Most project documents are designed based on templates. Some of the standard formats which are accepted worldwide are User Requirements Specification (URS), System Requirements Specification (SRS), System Test Cases (STC), User Acceptance Testing (UAT), Defect Track Log (DTL).

This paper describes features and methods for document comparison with templates and score generation. The uploaded documents are categorized based on the research area. The approach described here exploits the results of recent progress in information extraction to compare a document with its template and calculates a score. Clustering the documents is an important task in applications that analyze the template structure of a document. Using keyword Extraction algorithms, documents are tagged under appropriate categories to enhance the accuracy of search results for the project documents. Thus the documents in the repository serves the purpose of reference for students who are searching for projects under a specific area. This research would also help the project mentors to check the template matching of the documents submitted by students. Also our system helps students who are seeking new project, faculty and other academicians to get involved in ongoing projects and also to obtain ideas in their respective research domain.

Keywords : Parsing; XML; DTD; Tagging; Clustering; Project Document; Report; Stemming; Tagging; TF-IDF.

1. INTRODUCTION

Software projects are part of every computer science under graduate as well as post graduate courses. In each academic year hundreds of projects are carried out. The project deliverable documents must adhere to a specific standard template. The template differ from each genre of document types like System Requirements Specification (SRS), System Test Cases (STC) etc. For a project mentor it is impractical to manually verify the numerous project documents uploaded. This template based checking algorithm ensures the correctness of project documents. In the field of information retrieval, extraction of salient words as well as the meta-data is now possible. The focus of the work described here is to provide a method for checking whether a document follows its template format. Here, read the document and extract the document properties and generate XML files correspondingly for both document and also the template. That is at the end the comparison is between the XML documents.

This pooled document becomes the reference for future students and faculty guiding in various projects. For the proper use of this reference, categorization and clustering of the documents is very much relevant. In categorizing the documents, there are various algorithms such as KEA (Keyword Extraction Algorithm) [4] algorithm, MAUI

* Department of Computer Science and Applications, Amrita School of Engineering Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India thusharamg@am.amrita.edu

** Department of Computer Science and Applications, Amrita School of Engineering Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India nikhildominic007@gmail.com

(Multi-Purpose Automatic Topic Indexing) [5] algorithm, etc. In our proposed method we have adopted MAUI algorithm in a modified form for tagging the project documents. Tagging of documents helps to manage the documents for future reference.

2. BACKGROUND STUDY

This section of the paper represents the existing works and researches done on various document processing and content extraction. These studies mainly aim at improving the accuracy in document extraction which is followed by template generation and score calculation.

X-Diff[1] is an algorithm that integrates XML structure characteristics with normal tree-to-tree correction mechanism. The formula is analyzed and compared with XyDiff, a printed XML diff formula. Capturing visual similarity between different documents [2] came to popularity among researches. From the aspect of data retrieval, these options and algorithms is used to answer queries primarily based on layout similarities. Whitespace characters are used as layout delimiter in segmentation scheme. It is appropriate for machine-printed Manhattan layouts. Each layout image is divided into an m×n grid and that every cell of the layout grid as is referred to a bin. A bin is called a text bin if at least half it overlaps a text block. Else it is referred as a white space bin. Hence the distance computations, row alignment etc. are calculated using the above bins.

A PDF [3] document page is segmented into completely different logical structure regions like text blocks, picture blocks, vector graphics blocks and compound blocks. Identify each component and is resulted as an XML file. Using KEA algorithm [4], keywords are extracted automatically from text by applying Naive Bayesian algorithm. It is done in two stages. Training and Extraction. Both stages use Naive Bayesian machine learning algorithm for candidate keyword selection from input document. The training stage creates a model using the training set of documents. The extraction stage identifies the relevant keywords and uses the model created during the training phase to select the best keywords.

MAUI [5] algorithm is an enhanced version of KEA algorithm. MAUI stands for Multi-purpose automatic topic indexing. First the documents are automatically tagged with state-of-the-art keyword extraction algorithm, and later they use a new approach named MAUI. The main difference is that MAUI uses semantic information extracted from Wikipedia.

3. METHODOLOGY

This paper proposes an ideal system which helps project coordination by automated template checking and tagging the project documents for easy management. The implementation of this system starts with the extraction of template and document XML structure. Later, based on the standard templates, project documents like SRS, STC, etc. are examined. This reduces the manual work of project mentors. This algorithm generates a score based on the comparison done with the properties extracted.

Extraction of keyterms from the project documents for the purpose of tagging is done using the idea of MAUI [5] algorithm. MAUI algorithm is used for automatic topical indexing based on the KEA system. Here it is used for automated tagging of project documents.

A. Template Extraction

The project mentor chooses a relevant template for the project group. The template contains the titles, headings and other mandatory contents which should present in the document. And also the template follows a specific font size and font type rules for each section. The document submitting by the students should also follow the same rules. This template document is uploaded and the fields like main heading, font size, font type and content font size, font type etc are extracted. These fields are the key element of the template. To read a word document (.DOC) and grab the properties, Apache POI package can be used. Apache POI, is a project run by the Apache Software Foundation. It provides Java libraries which allows for reading and writing files in Microsoft Office formats. MS Word, PowerPoint and Excel are the formats supported by this package. Here Apache POI package is used to read the word documents and extract the data and its properties.

The output of the WordReader is mapped into a two-dimensional string array, in which each row contains paragraph from the document. That is each row in the string array is a set of characters from the document within a paragraph. Identifying the headings from these arrays is to be done so that a section of information, that is a heading followed by the contents can be identified. It is assumed that a row with less than 50 characters is a heading. Even if the assumption may be false in some cases, it can be eliminated on the next phase by deselection of the undesired ones.

Once a heading is identified the contents which come under this has to be identified. For this purpose, the location of the next heading need to be found, since all the rows which comes after a heading till the next heading would be the content. Because, all the paragraphs visiting on the way until a next heading is found, will be the contents. Another constraint to be considered is the content followed by the last heading. The contents which come under the last heading will not necessarily have a heading that follows them. Here to solve this problem at each iteration it is checked that whether it reached the length of the array. If so, the characters in between the heading and the last row is considered as content.

On successful completion of the parsing, each of them is traversed to recognize the properties such as font type, font size etc. and an XML file is generated.

B. Document Extraction

Once the template is extracted and the XML is generated, the sample document is to be parsed in the similar way. An XML should be generated with same schema definitions as of the template XML. Here the challenge is the unpredictable processor load, as the huge size and content rich documents are led for parsing. To avoid this, the document should be formatted. Images and rich graphic elements need to be removed. After the formatting of this document, the XML is generated in a similar manner.

C. XML Comparison

After the template and document XML generation, comparison is done between XML files. Comparing semi-structured documents has gained increasing attention since these documents are light weight and easy to use, even though from different perspectives like the purpose of characterizing a document with regard to a given DTD.

Here the priority goes to the template XML. Based on the template XML the document XML is evaluated. Headings from the template XML is taken and the document XML is traversed top to bottom for similar heading. If a similar heading is found then the heading number is stored temporarily. Then next step is to check for the similarity in properties like font size, font type, content font size, content font type.

D. Score Calculation

A variable for score is maintained throughout the comparisons. A score of 1 for every properties. That is one for each font size, font type, content font size, content font type. Thus a score of maximum 5 for each section. Then the total score of the document is calculated using (1).

(no. of headings in template) * 5 (1)

Percentage of the calculated score obtained from xml comparison with the total score (1) is calculated. Project mentor can set a threshold value there by deciding whether or not to accept the document. A report can be generated from this result which shows the percentage of documents passed the threshold.

E. Tagging

After successful validation of a project document, it is automatically tagged. This tags provides effective reference for future project groups. Tags are the keywords or tokens which are given for each documents. Keywords are the metadata which provides general information concerning the content of a document. They are the set of possible words by which a topic is searched by a user. Tagging is the way by which a project document is given a token of identification with the tag repository. That is, each uploaded document is mapped to a set of tags.

F. Learning and Extraction

For implementing tagging, the very first step is to identify the possible search keyphrases for each thrust areas. This is done by using some ideas of MAUI algorithm for extracting keywords from the valid documents that are to be tagged. MAUI algorithm leverages on features like TF-IDF calculation, calculation of position of the first occurrence, node degree, phrase length, keyphraseness, Wikipedia-based keyphraseness etc. Normally the thrust areas are considered as the terminology of these tags, for instance data mining, image processing, etc.

- **Tags from document title :** In most cases the document title contains the keywords that are most significant and accurate. The first phase of keyword extraction begins here. All the title keywords filtered by removing stopwords are considered to be the first set of keywords for the document.
- **Tokenization and stopwords removal and stemming :** The document is split word by word called tokens. This list of tokens is then subjected to stopword removal process. Stopwords are most common words in a language like 'the', 'is', 'at', 'which', 'on', etc. After tokenization, stemming process is done. Root words of these token is derived using stemming process. Example, the words 'processing', 'processed', 'processes', all these have the root word 'process'. In pure MAUI algorithm the process of tokenization, stopwords removal and stemming are separate processes. Doing these three processes in a single step will improve the through put of the algorithm. Each words identified in the process of tokenization is considered only if the word is not in the stopwords list. If it is not in the list then the word is subjected to stemming process. Thus each token stored into the token list are non stopwords and root words.
- **Wikipedia-based keyphraseness** is the occurrence of a token as a link in the Wikipedia corpus. A ratio of number of such links with the total number of links in that area is found. This ratio is multiplied with the frequency of the word in the document. This process gives keywords related to the document which are not present in the document.
- **TF-IDF** calculates a score with the frequency of a word in the document and the inverse occurrence frequency. For rare phrases the score will be high. Thus all the words with low TF-IDF will be significant. Keywords with a score less than the average is selected and marked as the keyword. Following is the mathematical formula for finding TF-IDF weights for each extracted keyterm.

  For a key, t and a document d, in a document corpus, D then,

$$\mathrm{TF}(t) = f(t, d) \tag{2}$$

$$\mathrm{IDF}(t) = \log\left(\frac{|\mathrm{D}|}{f(t, \mathrm{D})}\right) \tag{3}$$

$$\mathrm{TF} \times \mathrm{IDF} = \mathrm{TF}(t) \times \mathrm{IDF}(t) \tag{4}$$

where, d is an element of D. $f(t, d)$ the frequency of that with the total number of terms in the document, Term Frequency (TF) finds the highest weighed term that occurs in a document and Inverse Document Frequency (IDF) gives a measure of the frequency of a term across all documents present.

The modified MAUI steps are as follows:

1. Read document
2. Copy the whole text into a .txt file
3. The text file created in step 2 is read word by word where whitespace is the delimiter.
   Let 'currentstring' is the word which is processing currently and 'nextstring' is the string followed by the current string.
4. If (currentstring not in stopword list)
   currentstring is considered as token. The token is stemmed and added to token list
   Else
   currentstring is ignored
   currentstring = nextstring
   End If

5. If(nextstring is null)
   End
   Else
   Goto step 4
   End If

This specific tag is henceforth linked with the documents that are having these keyphrases associated. This enables the search option provided in this paper using tagging feature as discussed. So now we gathered and saved those keyphrases separately under corresponding tags which are based on the thrust area. Maximum number of keywords should be collected so that searching become more accurate. Also more the number of tags means more the success rate of relevant search result. *Table2* illustrates the outcome after applying MAUI on few valid documents related to various thrust areas. We obtained corresponding keyphrases under each trust areas and is afterward linked under suitable tags.

G. Searching

From the student's perspective, this feature focuses on a reference bucket. It has a large data pool which contains the different types of project documents of previous projects. Finding out a right project is difficulty for a student in the beginning of the project. Each faculty is focused on specific research/thrust areas and have different project experiences. With the search features, the students can easily find out project documents for reference.

All the uploads done by each project teams are stored in the document repository. Each of them are associated to their corresponding tags automatically. Once a search query is initiated from a user, it is first cross checked with the tag repository. If a tag is matched then all the documents which are mapped onto that tag is listed as search result.

For the convenience of the user, a snippet from the document is given along with each search results. It helps to choose the accurate document.

4. RESULTS AND ANALYSIS

The system would help the faculty to check the template matching of the documents submitted by students, and for students to refer related project documents as a repository where they can find sample project documents. This experiment is based on automating the manual process carried out by students and staff. Automatically tag the project documents using modified MAUI algorithm.

<HEADING>
<no>1</no>
<head> Software Requirements Specification </head>
font> Adobe Caslon Pro
<size> 32 </size>
<ctype> nill </ctype>
<csize>0 </csize>
</HEADING>
<HEADING>
<no>2</no>
<head> Introduction </head>
font> Adobe Caslon Pro
<size> 14 </size>
<ctype> nill </ctype>
<csize>0 </csize>

```
</HEADING>
<HEADING>
<no>3</no>
<head> Purpose </head>
<font> Adobe Caslon Pro </font>
<size> 14 </size>
<ctype> Adobe CaslopPro </ctype>
<csize>11 </csize>
</HEADING>
```

Fig. 1. Sample document XML output.

Fig.1 Illustrates a portion of output XML file. This file is generated using the fields from a sample project document. All XML files follow the same schema definition. The comparison process is done after generating similar XML for both document and the template files.

Table 1. Sample XML comparison result.

No	*Heading*		*Font Name*		*Font Size*		*Content Font Name*		*Content Font Size*	
1	Abstract	√	Times New Roman	√	14	√	Times New Roman	√	12	√
2	Introduction	√	Times New Roman	√	14	√	Times New Roman	√	12	×
3	Literature Review	√	Times New Roman	√	14	√	Times New Roman	√	12	√
4	Methodology	√	Times New Roman	√	14	√	Times New Roman	×	12	×
5	Results and Analysis	√	Times New Roman	√	14	√	Times New Roman	√	12	√

Table1. lists the result of document after comparing. It shows the properties of the document which matches with the template with a tick symbol (√) and the fields which fails to match with a cross (×). It helps to identify and rectify the text accurately.

Fig. 2. Final result with percentage of similarity.

Fig.2 depicts the percentage of similarity of the document with the template. Also it checks the threshold and rates the document. Mostly 80% or more is accepted. So the threshold value is set to 80. All the documents with score less than 80 is rejected.

Table 2. TF × IDF Calculated on sample set of documets

Project Documents	*Keyword*	*TF*	*IDF*	*TF × IDF*
Doc 1	graphical	0.0170	0.0000	0.0000
	prolog	0.0120	0.0969	0.0009
	regress	0.0022	0.3010	0.0006
	graph	0.0011	0.3010	0.0003
	program	0.0011	1.0000	0.0011
	traces	0.0056	0.6989	0.0039
Doc 2	mine	0.0170	0.0000	0.0000
	classif	0.0102	0.0969	0.0009
	associ	0.0022	0.3010	0.0006
	summari	0.0004	1.0000	0.0004
Doc 3	extract	0.0201	0.3010	0.0060
	predict	0.0057	0.2218	0.0012
	mine	0.0129	0.0000	0.0000
	kdd	0.0129	0.2218	0.0028
Doc 4	mine	0.0112	0.0000	0.0000
	classif	0.0117	0.0969	0.0011
	associ	0.0099	0.3010	0.0298
	C45	0.0067	0.5228	0.0035
	confid	0.0063	0.3979	0.0025

Table 2. Lists out the TF-IDF value of sample set of four documents. The value clearly shows that the keywords with low TF-IDF is more relevant to a document.

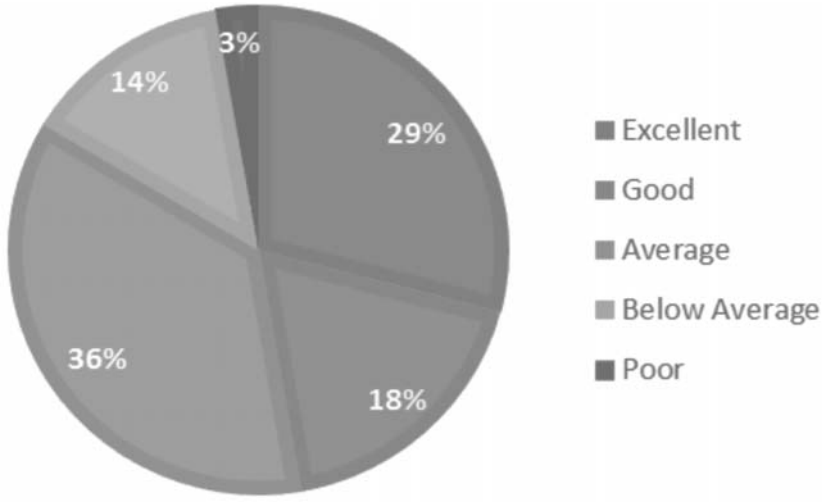

Fig. 3. Project document validation report of an academic year.

5. CONCLUSION

Introducing template based checking algorithm for the documents submitted by the students makes it easy for the project mentors to evaluate the documents in a few minutes with less strain. Algorithms that help in identifying the percentage of similarity with the template helps the above come true. The system specifically shows where a document violates its template structure. This helps in the correction of the document. Also it calculates a score which a project mentor can use to decide whether or not to accept the document initially.

Automatic tagging of documents using modified MAUI algorithm helps in faster processing. Automatic document tagging enhances the searching accuracy with relevant search results. This helps the students to find sample project documents for reference. Thus our system proves to be an effective aid for both faculty and students.

6. ACKNOWLEDGMENT

We would like to express our sincere gratitude to the faculty of Department of Computer Science and Applications of Amrita Vishwa Vidyapeetham, Amritapuri for providing help and guidance.

7. REFERENCES

1. Yuan Wang David, J. DeWitt Jin-Yi Cai, "X-Diff: An effective change detection algorithm for xml documents," Proceedings of the 19th International Conference on Data Engineering, 2003.
2. Jianying Hu, Ramanujan Kashi, Gordon Wilfong, "Comparison and classification of documents based on layout similarity", Information Retrieval 2, 227–243 (2000),
3. Hui Chao & Jian Fan, "Layout and content extraction for PDF documents", Hewlett-Packard Laboratories, Document Analysis Systems VI, 6th International Workshop, DAS 2004, Florence, Italy, September 8-10, 2004.
4. Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., Nevill-Manning, C. G. (1999, August). "KEA: Practical automatic keyphrase extraction", In Proceedings of the fourth ACM conference on Digital libraries (pp.254-255). ACM.
5. Olena Medelyan, Eibe Frank, Ian H. Witten, "Human-competitive tagging using automatic keyphrase extraction", EMNLP '09 Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Volume 3.
6. Priyanka G Pillai, Jayasree Narayanan, "Question categorization using SVM based on different term weighting methods", Amrita Vishwa Vidyapeetham, Amritapuri, International Journel on Computer Science and Engineering(IJCSE), Vol.4 No. 05 May 2012.