DOCUMENT CLASSIFICATION
Combining Structure and Content

Samaneh Chagheri, Sylvie Calabretto
Université de Lyon, CNRS, LIRIS UMR 5205, INSA de Lyon, 7 Avenue Jean Capelle, Villeurbanne, France

Catherine Roussey
Cemagref, Campus des Cézeaux, Clermont Ferrand, 24 Avenue des Landais, Aubière, France

Cyril Dumoulin
27, rue Lucien Langénieux, Roanne, France

Keywords: Document classification, Document structure, Technical document, Support vector machine, Vector space model.

Abstract: Technical documentation such as user manual and manufacturing document is now an important part of the industrial production. Indeed, without such documents, the products can neither be manufactured nor used according to their complexity. Therefore, the increasing volume of such documents stored in the electronic format, needs an automatic classification system in order to categorize them in pre-defined classes and to retrieve the information quickly. On the other hand, these documents are strongly structured and contain the elements like tables and schemas. However, the traditional document classification typically classifies the documents considering the document text and ignoring its structural elements. In this paper, we propose a method which makes use of structural elements to create the document feature vector for classification. A feature in this vector is a combination of the term and the structure. The document structure is represented by the tags of the XML document. The SVM algorithm has been used as learning and classifying algorithm.

1 INTRODUCTION

The term "technical documentation" refers to different product-related documents such as user manuals and product specifications. The increasing volume of these documents and their important role in the product life cycle requires an automatic classification system in order to categorize them in the proper pre-defined classed and to facilitate the information retrieval. The technical documents are strongly structured and they have more tables and schemas than the general document collections. However, the traditional classification systems use only the document content and ignore its structural elements for classification. But the continuous growth in the number of structured documents as XML documents has increased the need for developing the classification systems based on the document structure. These systems extract the available structural elements in the document, and apply them for document representation in order to improve the classification accuracy. The document structure is represented as the tags of the XML document. However, much of the information is contained in the text fields not just in tag labels; therefore, devising a method which exploits the structure and the content of the document is desirable.

In this article, we have proposed a document representation method which is an extension of the vector space model of Salton (Salton, 1968) adjusting the calculation of the *tf*idf* by considering the structural element instead of the entire document. The document is represented as a tree in which nodes are structural elements, and leaves are the document textual parts. The rest of the article is organized as follows. We present in section 2 the different approaches proposed in the literature to XML document classification. Basic methods for classification are described in section 3. Our proposal on document representation and feature vector construction is explained in section 4. Section 5 describes the experiments and results. And finally, section 6 presents the conclusion and future works.

2 RELATED WORKS

The continuous growth in XML documents has caused different efforts in developing classification systems based on document structure. Document representation has to be done before classification process. The representation models can be divided into three groups: the first group are the models which do not consider the structure of document. These works focus on representing the document content as a bag of words to classify them. They are the most studied classification methods. The second group are the models which take into account only the structure of a document in order to classify them without considering the document content. For example (Wisniewski et al, 2005) use Bayesien model to generate the possible DTDs of a documents collection. A class represents a DTD. They are interested only on document structure for classification. **(**Aïtelhadj et al, 2009) and (Dalamagas et al, 2005) have also classified the documents by only document structure trees similarities. Finally, the third group is composed of the models which consider both structure and content of XML documents in representation.

Mostly the classification systems use vector space model for document representation, the difference is based on selecting vector features and their weight computation. In (Doucet and Ahonen-Myka, 2002) each vector feature can be a word or a tag of XML document. The *tf*ief* (Term Frequency * Inverse Element Frequency) is used for calculating the weight of the words or the tags in documents. In (Vercoustre et al, 2006) a feature can be a path of the XML tree or a path following by a text leaf. The term weight is based on *tf*idf*. This allows taking into account either the structure itself or the structure and content of these documents. (Yi and Sundaresan, 2000) have also used a vector model containing terms or XML tree paths as the vector elements. Ghosh (Ghosh and Mitra, 2008) has proposed a composite kernel for fusion of content and structure information. The paths from root to leaves are used as indexing elements in structure kernel weighted by *tf*idf*. The content and structure similarities are measured independently. A linear combination of these kernels is used finally for a content-structure classification by SVM. (Wu and Tang, 2008) propose a bottom up approach in which the structural elements are document tree nodes and the leaves are textual section of each element. First the terms in leaf nodes are identified, and their occurrences are extracted and normalized. Then the key terms are substantiated with the structural information included in the tags by the notion of key path. A key path ends at a leaf node that contains at least one key term for a class. By using the key terms set and key path the similarity is computed between new documents and class models. (Yan et al, 2008) propose a document representation by the vectors of weighted words, a vector for each structural element in a document. In this method a weight is associated to each structural element according to its level in the document tree. Then the weight of words is calculated based on its frequency inside the element and the importance of this element. (Yang and Wang, 2010) (Yang and Zhang, 2008) have proposed an extension of the vector model called the Structured Link Vector Model (SLVM). In their model a document is represented by a set of term vectors, a vector for each structural element. The term weight is calculated by term frequency in each document element and the inverse document frequency of term.

The model that we propose belongs to the third group, constructing the document feature vector by structure and content.

3 DOCUMENT CLASSIFICATION

Classification can be divided in two principal phases. The first phase is document representation, and the second phase is classification. The standard document representation used in text classification is the vector space model. The difference of classification systems is in document representation models. The more relevant the representation is, the more relevant the classification will be. The second phase includes learning from training corpus, making a model for classes and classifying the new documents according to the model. In this section we are going to explain the vector space model followed by a presentation of the SVM classification algorithm which is used in our approach.

3.1 Vector Space Model

The Vector Space Model (VSM) proposed by Salton (Salton, 1968) is a common technique for document representation in classification. In this model each document is represented as a vector of features. Each feature is associated with a weight. Usually these features are simple words. The feature weight can be simply a Boolean indicating the presence or absence of the word in document, its occurrence number in document or it can be calculated by a formula like the well known *tf*idf* method.

So, the feature vector of document d of collection D with n distinct terms is represented as follows:

$$d_d= [w_{1,d}, w_{2,d}, \ldots, w_{n,d}] \quad (1)$$

$$w_{i,d}= tf_{i,d}* idf_i \quad (2)$$

$w_{i,d}$ is the weight of term i of document d, where tf is the frequency of term i in document d and idf_i = log (|D|/|D_i|) is inverse document frequency, |D| is the total number of the documents in collection D, and |D_i| is the number of documents in collection containing the term i.

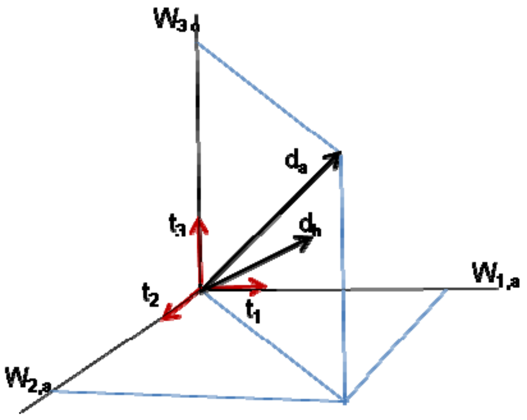

Figure 1: A document feature vector.

But in classical vector space model, document structure is not considered. For taking into account the notion of structure in document representation an extension of VSM seems promising.

3.2 Support Vector Machine

The Support Vector Machine (SVM) proposed by Vapnik (Cortes and Vapnik, 1995), is a supervised learning algorithm that can be applied to classification. It is a binary linear classifier which separates the positives and negatives examples in a training set. The method looks for the hyperplane that separates positive examples from negative examples, ensuring that the margin between the nearest positives and negatives is maximal. The effectiveness of SVM is superior to other methods of text classification. SVM makes a model representing the training examples as the points in a dimensional space separated by the hyperplane, and it uses this model to predict a new example belongs to which side of this hyperplane. The examples used in searching the hyperplane are no longer used and only these support vectors are used to classify new case. This makes a very fast method.

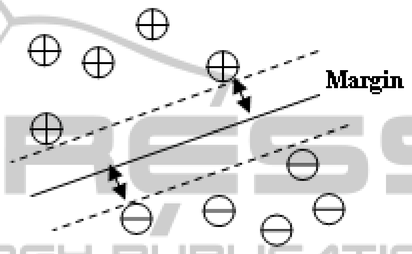

Figure 2: Maximum-margin hyperplane.

4 GENERAL OVERVIEW OF OUR APPROCH

In this article we propose a combination of content and structure of XML document in the vector model for document representation. In our vector, each feature is a couple of (tag: term). Tag corresponds to a structural element in XML document. XML document is represented as a tree in which the nodes are the XML tags and the leaves are the textual part in each document element. To construct our features we apply two processes. First, a process is done on the logical structure document in order to construct the aggregated tree of XML document. Secondly, a lexical process is done on document content to select the most informative terms. These processes aim to reduce the computational complexity and improve system performance. Then the weight of features is computed. Finally, the document vectors are used as the inputs of classification system.

4.1 Document Tree Construction

We consider the XML document as a tree in which the nodes are tagged by structural labels like title, chapter, etc. The arcs of this tree represent the inclusion relation between nodes, and the leaf nodes contain the document text. The depths of nodes in document tree are important. Thus we consider that two nodes with the same label localized at different depths are two different structural elements. The structural element represents a node type.

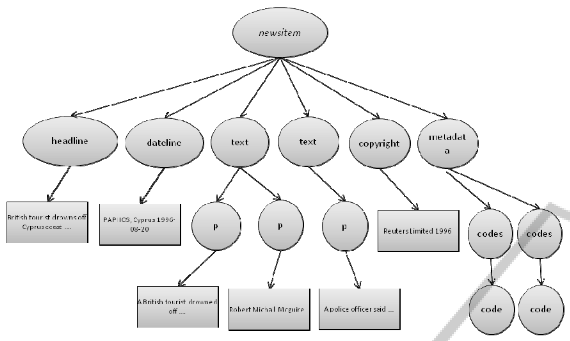
newsitem
headline
dateline
text
text
copyright
metadat a
British tourist drowns off Cyprus coast ....
PAP ICS, Cyprus 1996-08-20
p
p
p
Reuters Limited 1996
codes
codes
A British tourist drowned off ....
Robert Michal Mcguire
A police officer said ...
code
code

Figure 3: XML document tree.

We modify document tree organization by aggregating the nodes with the same label and localized at the same depth. For example, all paragraphs in a section are aggregated to a single node "paragraph" which contains all terms of these paragraphs. We take into account only the leaf nodes which contain text. We assume that all documents of collection satisfy the same logical structure. So, a label becomes an adequate identifier of the structural element. Also, we filter some tags which are not representative for document semantic based on a tag list made manually.

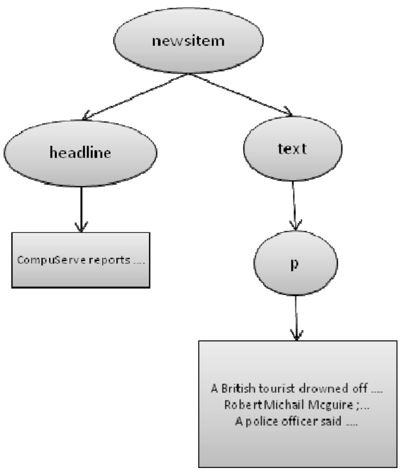
newsitem
headline
text
CompuServe reports ....
p
A British tourist drowned off ....
Robert Michail Mcguire ;...
A police officer said ....

Figure 4: XML Document aggregated tree.

4.2 Feature Extraction

After constructing the aggregated tree, we extract the terms in each node. We apply a series of linguistic analysis. First we extract the terms lemma and their part of speech using TreeTagger. Secondly we filter the words, just nouns and verbs are kept and the other words are removed. Then the words are replaced by their lemma. The result of such analysis is called term. Therefore, each feature is made by combining the node label and the term, for example (title: technical) or (p: cleaning). This feature represents the document structure and content. Our hypothesis is that a term which appears in two different structural elements should be considered as two different features. Also, its weight is different. Moving down in document tree, form root to leaf, decreases the elements importance. For example the word "classification" appears in the document title and in one paragraph composes two different features: (title: classification) and (paragraph: classification). The weight of the first feature is more important than the second.

4.3 Weight Computation

We assume that terms occurring in different structural elements have different importance. For calculating the weight of features we use an extension of traditional tf*idf on structural element level instead of document level. We also take into account the importance of the structural element in which the term has appeared. We assume that the deeper a node is in the document tree, the less it will be important. Therefore, the weight of feature will be calculated as below:

$$\mathrm{W_{i,e,d}} = \mathrm{TF_{i,e,d}} * \mathrm{IDEF_{d,e}} * \mathrm{IED_e} \quad (3)$$

$$\mathrm{IDEF_{d,e}} = \log \frac{|\mathrm{D_e}|}{|\mathrm{D_{e,i}}|} \quad (4)$$

$$\mathrm{IED_e} = \log\left(\frac{\mathrm{L_d} + 1}{\mathrm{l_{d,e}}}\right) \quad (5)$$

Where *i, e,* and *d* represent respectively the term i, the structural element e, like "paragraph" and the document d in the collection.

The Term Frequency $TF_{i,e,d}$ is the number of occurrences of the term *i* in structural element type *e* inside the document *d*. $IDEF_{d,e}$ is the Inverse Document Element Frequency with $|D_e|$ as the number of documents in the collection having a node of type *e*, and $|D_{e,i}|$ as the number of documents having node *e* containing the term *i*. IED_e is the Inverse Element Depth that represents the importance of the structural element *e* in the document. Where L_d is the depth of the document tree, and $l_{d,e}$ is the depth of the node *e* in this document.

After extracting all features in documents and calculating their weight, document vector is constructed and SVM is used for learning and classifying.

5 EXPERIMENTATION AND RESULTS

We have developed a prototype for the document classification system by implementing the algorithm described in the previous section. The implementation is on java with some functionality like extracting the document schema and structure, extracting the words and generating the combined feature, calculating the feature weight and constructing the document feature vector. The TreeTagger and GATE Tokeniser have been used for word processing. The SVM^{light} (Joachims, 1999) has been used for learning and classifying the documents. It is an implementation of SVM algorithm. This algorithm has scalable memory requirements and can handle problems with many thousands of support vectors efficiently.

5.1 Test Collection

Experiments were performed on the Reuters Corpus Volume 1 (RCV1) which includes over 800,000 English language news stories. Reuters is a leading global provider of financial information, news and technology to financial institutions, the media, businesses and individuals. The stories in this collection are formatted using a consistent XML schema. Reuters collection has been used by many researchers in the field of information retrieval and machine learning. All the stories in RCV1 have been coded for topic, region (geography) and industry sector and they are multiclass. Another test collection is selected from INEX XML document mining collection.

In order to do a mono classification we have used the first topic code in documents as the class of documents. Also the unclassified documents are removed from collection. We have used 800 documents of Reuters collection by considering 400 positive and 400 negative examples for a topic class called "GCAT". After analyzing the structural elements in documents, we have selected the header and paragraphs tags in stories. These tags seem more representative for document topic. Other tags are removed from document tree. TreeTagger and Tokeniser of platform GATE has been used to extract the terms. In INEX collection the documents are single label for mono classification and the selected structural elements are name, figure caption, table, section and paragraph.

5.2 Experimentation

We have performed two experiments on test collections by using SVM. The first one called "content and structure" is an implementation of our proposed method in which the document vector is made by tag and term, and where the feature weight is also calculated using our proposed formula. The second one called "content only" uses a vector of simple terms weighted by standard *tf*idf*. The application has been implemented in java except SVM^{light} algorithm which is on C. K-fold Cross validation has been used as the method for evaluating the system performance.

The results of the classifications are shown in table 1. Precision and recall as the most widely used metrics for evaluating the correctness of classification have been used. Another used metric is F-Measure that combines precision and recall. The results demonstrate that including structure has improved the classification accuracy in Reuters collection which is a homogenous collection by a limited and homogenous structural elements set. But in INEX Collection with heterogeneous structural elements the results shows a worse recall. Some modification is performing in order to manage the heterogeneous collections with different XML schemas.

Table 1: Classification results.

Vector features	collection	Precision	Recall	F-measure
Content & structure	Reuters	0.96	0.92	0.94
Content only		0.93	0.85	0.89
Content & structure	Inex	0.98	0.48	0.65
Content only		0.98	0.60	0.75

6 CONCLUSIONS AND FUTURE WORKS

In this article we have proposed a model for the XML document representation in order to classify them in the pre-defined classes. We proposed a combination of the structure and the content in document representation by the vector model. The features are the couples of *(tag: term),* all weighted by an extension of *tf*idf* taking into account the terms position in the documents tree. The document tree is constructed by filtering non relevant structural elements and aggregating the nodes having the same label and the same path in tree.

We have done the experiments on Reuters XML news collection and INEX XML mining collection. In Reuters collection all documents share the same logical structure. In future works we intent to improve our proposition in order to achieve a better performance on the heterogeneous technical document collection of Continew. Continew is a company that ensures the storage and security of the critical data and the technical documentation. The documents in this collection have been created by different authors using different styles to present their logical layout. So, we have to first detect the logical structure of the document, reconstruct the document structure and then use this structure for classification.

REFERENCES

Aïtelhadj, A., Mezghiche, M., & Souam, F. (2009). Classification de Structures Arborescentes: Cas de Documents XML. *CORIA 2009*, 301-317.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning* , 273-297.

Dalamagas, T., Cheng, T., Winkel, K.-J., & Sellis, T. (2005). Clustering XML Documents Using Structural Summaries. *EDBT* , 547-556.

Doucet, A., & Ahonen-Myka, H. (2002). Naive Clustering of a Large XML Document Collection. *INEX Workshop 2002* , 81-87.

Ghosh, S., & Mitra, P. (2008). Combining Content and Structure Similarity for XML Document. *ICPR* , 1-4.

Joachims, T. (1999). Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press* , 169-184.

Joachims, T. (1999). Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press* , 169-184.

Salton, G. (1968). *Search and retrieval experiments in real-time information retrieval.* (C. University, Ed.) 1082-1093.

Vercoustre, A.-M., Fegas, M., Lechevallier, Y., & Despeyroux, T. (2006). Classification de documents XML à partir d'une représentation linéaire des arbres de ces documents. *EGC 2006* .

Wisniewski, G., Denoyer, L., & Gallinari, P. (2005). Classification automatique de documents structurés. Application au corpus d'arbres étiquetés de type XML. *CORIA 2005 Grenoble* , 52-66.

Wu, J., & Tang, J. (2008). A bottom-up approach for XML documents classification. (ACM, Ed.) *ACM International Conference Proceeding Series; Vol. 299* , 131-137.

Yan, H., Jin, D., Li, L., Liu, B., & Hao, Y. (2008). Feature Matrix Extraction and Classification of XML Pages. *APWeb 2008 Workshops* , 210-219.

Yang, J., & Wang, S. (2010). Extended VSM for XML Document Classification Using Frequent Subtrees. *INEX 2009* , 441-448.

Yang, J., & Zhang, F. (2008). XML Document Classification Using Extended VSM. *INEX 2007* , 234–244.

Yi, J., & Sundaresan, N. (2000). A classifier for semi-structured documents. *KDD '00* , 340-344.