ELSEVIER

Information Processing and Management 40 (2004) 807–827

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Bayesian network model for semi-structured document classification

Ludovic Denoyer *, Patrick Gallinari

Laboratoire d'Informatique de Paris VI, LIP6, 8 rue du Capitaine Scott, 75015 Paris, France

Available online 17 June 2004

Abstract

Recently, a new community has started to emerge around the development of new information research methods for searching and analyzing semi-structured and XML like documents. The goal is to handle both content and structural information, and to deal with different types of information content (text, image, etc.). We consider here the task of structured document classification. We propose a generative model able to handle both structure and content which is based on Bayesian networks. We then show how to transform this generative model into a discriminant classifier using the method of Fisher kernel. The model is then extended for dealing with different types of content information (here text and images). The model was tested on three databases: the classical webKB corpus composed of HTML pages, the new INEX corpus which has become a reference in the field of ad-hoc retrieval for XML documents, and a multimedia corpus of Web pages.

Keywords: Statistical learning; Bayesian networks; Categorization; Structured documents; XML; Machine learning

1. Introduction

Document classification is used in many different contexts in information retrieval: document filtering, word sense disambiguation, document classification in hierarchies like those of Yahoo!, etc. This field mainly developed over the last ten years, using techniques originating from the pattern recognition and machine learning communities. Almost all classification techniques which have been proposed in recent years (e.g. neural networks, support vector machines, decision trees and decision lists, etc.) have been tested on this problem. All these methods do operate on flat text representations and do not consider text structure information. Some attempts have recently been made to relax the traditional word independence assumption. Denoyer, Zaragoza, and Gallinari (2001) for example consider a limited form of sequence information and use hidden Markov models for text and passage classification. The recent paper (Sebastiani, 2002) gives a very good survey of the literature on textual document classification. With the development of structured textual and multimedia documents, and with the increasing importance of structured

* Corresponding author.
E-mail addresses: ludovic.denoyer@lip6.fr (L. Denoyer), patrick.gallinari@lip6.fr (P. Gallinari).

0306-4573/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2004.04.009

document formats like XML, the document nature is changing. Structured documents usually have a much richer representation than flat ones. They have a logical structure. They allow the incorporation of additional information such as metadata and are often composed of heterogeneous information sources (e.g. text, image, video, etc.). The development of classifiers for structured content is a new challenge for the machine learning and IR communities. Since this is a new area there is not yet a consensus on what the main tasks and challenges of structured document classification are. A major change with structured documents compared to flat documents is the possibility to access document elements or fragments. Accordingly, a classifier for structured documents should be able to classify both full documents and document parts. It is also important to be able to make use of the different content information sources present in an XML document. A classifier should then easily adapt to a variety of different sources. A final requirement is that the system be able to scale with large document collections.

We propose here a new model for the classification of structured documents. It is a generative model based on Bayesian networks. Each document will be modelled by a Bayesian network, the size of which being proportional to the size of the document. Classification will then amount to perform inference in this network. The model is able to take into account the structure of the document and different types of content information. It also allows one to perform inference either on whole documents or on document parts taken in their context, which goes beyond the capabilities of classical classifier schemes. In this paper, the elements we consider are defined by the logical structure of the document. They typically correspond to the different components of an XML document. Different types of Bayesian models could be used for the documents. For keeping the computations to a reasonable complexity level and for allowing robust parameter estimation, we have to restrict ourselves to simple models exploiting local structural dependencies. We further show how these generative models can be turned into discriminant classifiers using Fisher kernels. Doing this, we lose part of the potential of the Bayesian network model. Compared to the latter, the Fisher kernel classifier does not offer a natural framework for classifying document fragments but increases the classification accuracy for full documents. It could also be trained for classifying predefined fragment types at a price of an increased complexity.

We first review previous work in Section 2, we then introduce structured documents in Section 3 and our core Bayesian network model in Section 4. We describe in Section 5 how to learn the network parameters from a document corpus. We introduce the Fisher kernels in Section 6. In Section 7 we show on an example how the model may be used with different types of content information. We then describe tests on three collections (Section 8): a classical benchmark where textual documents correspond to academic Web sites, a large corpus of XML documents and a large collection of Web sites where both the textual and image content are considered for classification.

2. Previous works

Handling structured documents for different IR tasks has recently attracted an increasing attention. However, it rapidly appeared that designing new information retrieval systems so that they can handle structured documents is far from trivial. Many questions are still open for designing such systems so that we are only in the early stages of this development. Most of the work in this new area has concentrated on ad hoc retrieval. Two recent Sigir workshops (2000 and 2002) where dedicated to this subject. A series of papers describing on-going research on different aspects of structured document storage and access, ranging from database problems to query languages and IR algorithms is available in a special issue of JASIST (Baeza-Yates, Carmel, Maarek, & Soffer, 2002). There is now a small but active community on this area. Most teams involved in this research gather around the recent initiative for the development and the evaluation of XML IR systems (INEX) which has been launched in 2002. Problems which are debated at INEX concern: indexing structured document, defining different types of "content and structure" queries

for structured documents, designing query languages, defining what type of relevant fragments should be retrieved, extending IR models or designing new models for semi-structured document access, defining new evaluation criteria (Fuhr, Govert, Kazai, & Lalmas, 2002).

Besides this mainstream of research, some work is also developing around other generic IR problems like clustering and classification for structured documents. For now, clustering has mainly been dealt with in the database community. This work has focused on structure clustering for indexing XML databases, and ignored the document content (Termier, Rousset, & Sebag, 2002; Zaki & Aggarwal, 2003). Structured document classification has also recently motivated some research. Since this is the focus of the paper, this is discussed in greater length below.

From a modeling perspective, there have been two main approaches for classifying structured documents. The first one concerns the majority of published papers. It makes use of different flat text classifiers operating on distinct document elements, these base classifiers are then combined for classifying the whole document. A second family attempts to design new types of classifiers adapted for structured documents. The first approach has mainly been developed for the categorization of whole HTML pages using the semantic of HTML tags. Quek (1997) for example combines three classifiers operating respectively on the textual information of a page, the page and section titles and hyperlinks. Results show that title and hyperlink information are relevant for the task and improve the performance compared to a simple flat text classifier. The approach by Yang, Slattery, and Ghani (2002) is similar: three classifiers respectively use linked pages textual information, HTML tags and metadata. Cline (1999) maps a structured document onto a fixed-size vector where each structural entity (title, links, text, etc.) is encoded into a specific part of the vector using classical tf-idf. A single classifier then processes this vector. In his experiments, he did not find any improvement compared to flat classifiers on the webKB corpus (webKB, 1999). In another work (Dumais & Chen, 2000) make use of the HTML tags information to select the most relevant part of each document. Chakrabarti, Dom, and Indyk (1998) propose to use the information contained in neighboring documents of an HTML pages. All these methods explicitly rely on the HTML tag semantic. For example, the models by Quek (1997) or Yang et al. (2002) need to ''know'' the semantic of the HTML tags, i.e. whether they correspond to a title, a link or a reference, etc. They are HTML dependent, they cannot adapt to more general structured categorization tasks where for example the tag semantic is not known in advance or can vary from a document collection to another. Most of these models rely on a vectorial description of the document and do not offer a natural way for dealing with document fragments. We will show in Section 4 that our model is not dependent of the semantic of the tags and is able to learn which parts of a document are relevant for the classification task.

The second family of models uses more principled approaches for structured documents. Yi and Sundaresan (2000) propose a vector representation for tree-like structured documents. They develop a probabilistic model for whole document classification. A characteristic of this model is that it makes use of local word frequencies which depend on the path from the document root to content nodes. They then face a very severe estimation problem for their local probabilities. They performed experiments on two small document collections, each with a very basic structure. Diligenti, Gori, Maggini, and Scarselli (2001) propose the hidden tree Markov model (HTMM). This is an extension of HMMs to tree like structures. One generative model is built per class. Each model has a fixed number of hidden variables responsible for the emission of the document content. For modeling a document, a hidden tree is built with the same structure as the document. Nodes in this tree correspond to hidden variables. Observations correspond to the textual content of the document and are emitted by the hidden variables. The probability of a document being emitted by a particular model (i.e. class) is the summation of the tree probabilities for all allowable configurations of the hidden variables. This is similar to HMMs summing over all possible state sequences. The textual content of document elements is encoded as a vector of 0/1 denoting the presence or absence of terms. The model is trained using EM. They performed tests on the webKB collection showing a slight improvement over Naive Bayes (1%). This model is close to ours in the sense that it makes use of tree like

probabilistic models, and uses somewhat similar simplifying assumptions for computing document probabilities. Their model was initially inspired for image classification and there are important differences with ours. We do not use hidden variables. This avoids summing over all the hidden variable values combinations over the document nodes. Since this is combinatorial, this is unfeasible for large documents even for a small number of hidden states. We also use a richer content description. Outside the field of information retrieval, some related models have also been proposed. The hierarchical HMM (HHMM; Fine, Singer, & Tishby, 1998) is also a generalization of HMMs. It has not been proposed to deal directly with structures but with sequences which exhibit multi-scale and recursive structures. Hidden nodes in this model emit sequences instead of symbols for classical HMMs. States may be activated recursively which allows for modelling nested subsequences. Compared to our model, the HHMM is aimed at discovering substructures in sequences instead of processing structured data.

Generative models have been used for document classification and clustering for a long time. The best known model is the Naive Bayes (Lewis, 1998). This is the simplest instance of a Bayesian networks. Naive Bayes extensions like the "Tree Augmented Naive Bayes" (TAN) designed for taking into account local dependencies between terms have also been tested for document classification (Koller & Sahami, 1997). They may offer slight improvements with respect to Naive Bayes in some cases. Probabilistic models with latent variables have been used recently for text clustering and classification by different authors. The general idea is to model via latent variables some hidden relations between textual elements. These models are usually trained with the EM algorithm. Vinokourov and Girolami (2001) train a generative model for building document hierarchies. They also propose to use a Fisher kernel derived from their generative hierarchy model for classifying the documents in the collection. Hofmann and co-workers proposed a series of latent variable models for different IR task. Hofmann proposed probabilistic latent semantic analysis (PLSA; Hofmann, 1999b) which is a generative model for analyzing co-occurrence data. In Hofmann (1999a) it is proposed to perform document clustering by considering simultaneously document similarity and word specificity for characterizing documents. Cai and Hofmann (2003) propose to use automatically extracted concepts together with classical word-based representations for classifying documents. All these models do operate on flat document representations. Using the same type of ideas, Blei and Jordan (2003) describe a series of models of increasing complexity for learning the correspondence between images or image regions and image captions. Their models may be used for modeling both the joint distribution of images and associated words, and conditional distributions of words given images. Modeling the latter conditional distributions allows the authors to perform tasks like automatic annotation or text based image retrieval. This is an original aspect of their work. Density models for image and captions operate on flat content representations. They perform experiments on the Corel image database where captions are composed of 2–4 words. This model has been designed for learning dependencies between two different representations of the same object, it does not handle structured representations. The focus of their work is on dependencies which are presumably much simpler than those present between the different elements of a large structured document.

Finally, Bayesian networks have been used for the task of ad-hoc retrieval both for flat documents (Callan, Croft, & Harding, 1992) or for structured documents (Myaeng, Jang, Kim, & Zhoo, 1998; Piwowarski, Faure, & Gallinari, 2002). This is different from classification since the information need is not specified in advance. The models and problems are therefore different from those discussed here.

3. Structured document

In the following, we will consider that a document is a tree where each node represents a structural entity. This corresponds to the usual representation of XML documents and this is also the classical structured document representation. A node in the tree will contain two types of information:

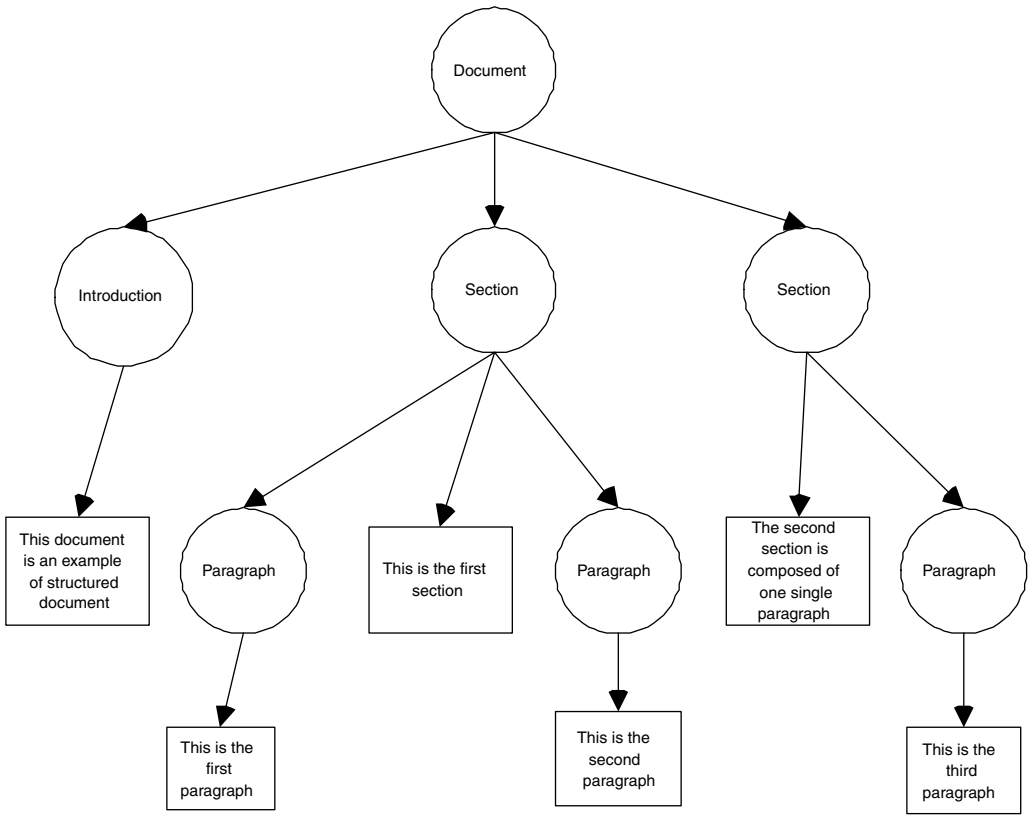


Fig. 1. A tree representation for a structured document. This document is composed of an introduction and two sections. The first section has two paragraphs and the second one has only one paragraph. Circle and Square nodes are respectively structural and textual nodes.

- A label information which represents the type of the structural entity. A label could be for example *paragraph*, *section*, *introduction*, *title*, etc. The set of labels depends on the documents corpora we are dealing with. Usually, for XML documents, these labels are defined in the DTDs.
- A content information: for a multimedia document this could be either text, image or signal. For example, for a node with label *paragraph*, the content will be the paragraph text.

We will talk of structural and content nodes for these two types of information.

Fig. 1 gives an example for a simple textual document: structural and content nodes are respectively indicated as circles and square nodes.

For simplification, we will describe our model by considering only textual documents. We will then show in Section 7 how it can be naturally extended for handling different types of content.

4. Modeling documents with Bayesian networks

We will now describe the probabilistic structured models used for the documents.

Let us first define the notations:

- Let C be a discrete random variable which represents a class from the set of classes $\mathscr{C}$.
- Let Λ be the set of all the possible labels for a structural node.
- Let V be the set of all the possible words. V^* denotes the set of all possible word sequences, including the empty one.
- Let d be a structured document consisting of a set of features $(s_d^1, \ldots, s_d^{|d|}, t_d^1, \ldots, t_d^{|d|})$ where s_d^i is the label of the ith structural node of d ($s_d^i \in \Lambda$), t_d^i is the textual content of this ith node ($t_d^i \in V^*$) and $|d|$ is the number of structural nodes. d is a realization of a random vector D. In the following, all nodes are supposed to have a unique identifier, indicated here as superscript i.

Documents will be modeled using Bayesian networks. This is a suitable framework for modeling the dependencies and relations between the different elements in a structured document. We will associate a network model to each document. Since we focus here on the logical document structure, each network will be defined according to the corresponding document structure. For our classification task, the network parameters will be learned on all the documents from the same class in the training set. Documents from the same class will then share their parameters and there is one set of such parameters for each class.

Different networks could be used for modeling a document, depending on the which type of relation we would like to take into account. We have only considered here the explicit document structure and we will not try to uncover any hidden or implicit structure between the document elements. With this in mind, some of the natural relations which could be modeled are: "is a descendant of" in the document tree, "is a sibling of", "is a successor of"—given a preorder visit of the document tree—, and combinations of these different possibilities. In Figs. 2 and 3 we give two examples of document models encapsulating different relations. There are two types of variables corresponding respectively to structure nodes (s) and content nodes (t). In the simplest one (2), the network structure is similar to the document tree structure. The network only encodes the "is a descendant of" relation. More complex networks using additional local structural dependencies may also be used, the second example (Fig. 3) makes use of a TAN network at each level of the tree. This model takes into account an ordering relation between structural siblings and sub-trees which is not the case for the first one. Although it might be desirable to model rich relations between document elements, there is as usual a trade-off between complexity and efficiency. In order to test the potential of different models for the classification task, we performed a series of tests with models encapsulating different types of relations. They did not show a clear superiority of one model over the others with respect to the classification performances. In the following, we will then focus on the simple tree like model illustrated by the example in Fig. 2. Note that the development we present for this model can be easily adapted for more complex ones. As far as we limit ourselves to models with regular local dependencies, and for which all structural nodes do have the same fixed number of parents, there is no fundamental difference for inference and learning. Of course the model expressivity might be very different. From now on, we will then consider tree like Bayesian networks. The network structure is built from the document tree, but need not be identical.

4.1. Tree-like model for structured document classification

For this model, we make the following assumptions:

- There are two types of variables corresponding to structure and content nodes.
- Each structure node may have zero or many structure sub-nodes and zero or one content node.
- Each feature of the document depends on the class c we are interested in.
- Each structural variable s_d^i depends on its parent $\mathrm{pa}(s_d^i)$ in the document network.
- Each content variable t_d^i depends only on its structural variable.

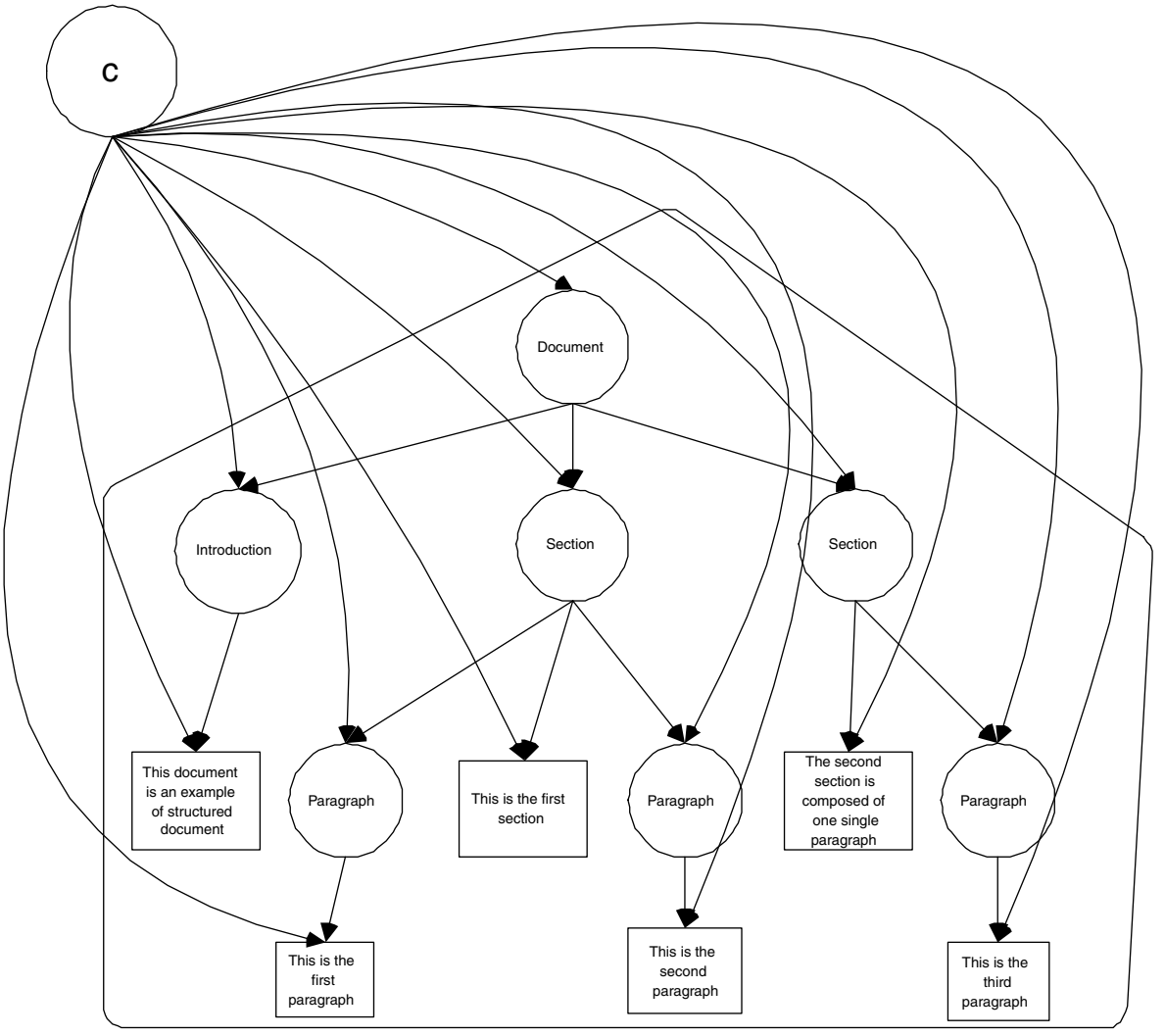


Fig. 2. The final Bayesian network encoding "is a descendant of" relation.

The generative process for the model corresponds to a recursive application of the following process: at each structural node s, one chooses a number of structural sub-nodes, which could be zero, and the length of the textual part if any. Sub-nodes labels and words are then sampled from their respective distribution which depends on s and the document class. The document depth could be another parameter of the model. Document length and depth distributions are omitted in our model since the corresponding terms fall out for the classification problems considered here.

Using such a network, we can write the joint probability

$$P(d,c) = P(c)\prod_{i=1}^{|d|} P(s_d^i|\mathrm{pa}(s_d^i),c)P(t_d^i|s_d^i,c) \tag{1}$$

Note that the textual part t_d^i could be empty since it belongs to V^* the set of all possible strings.

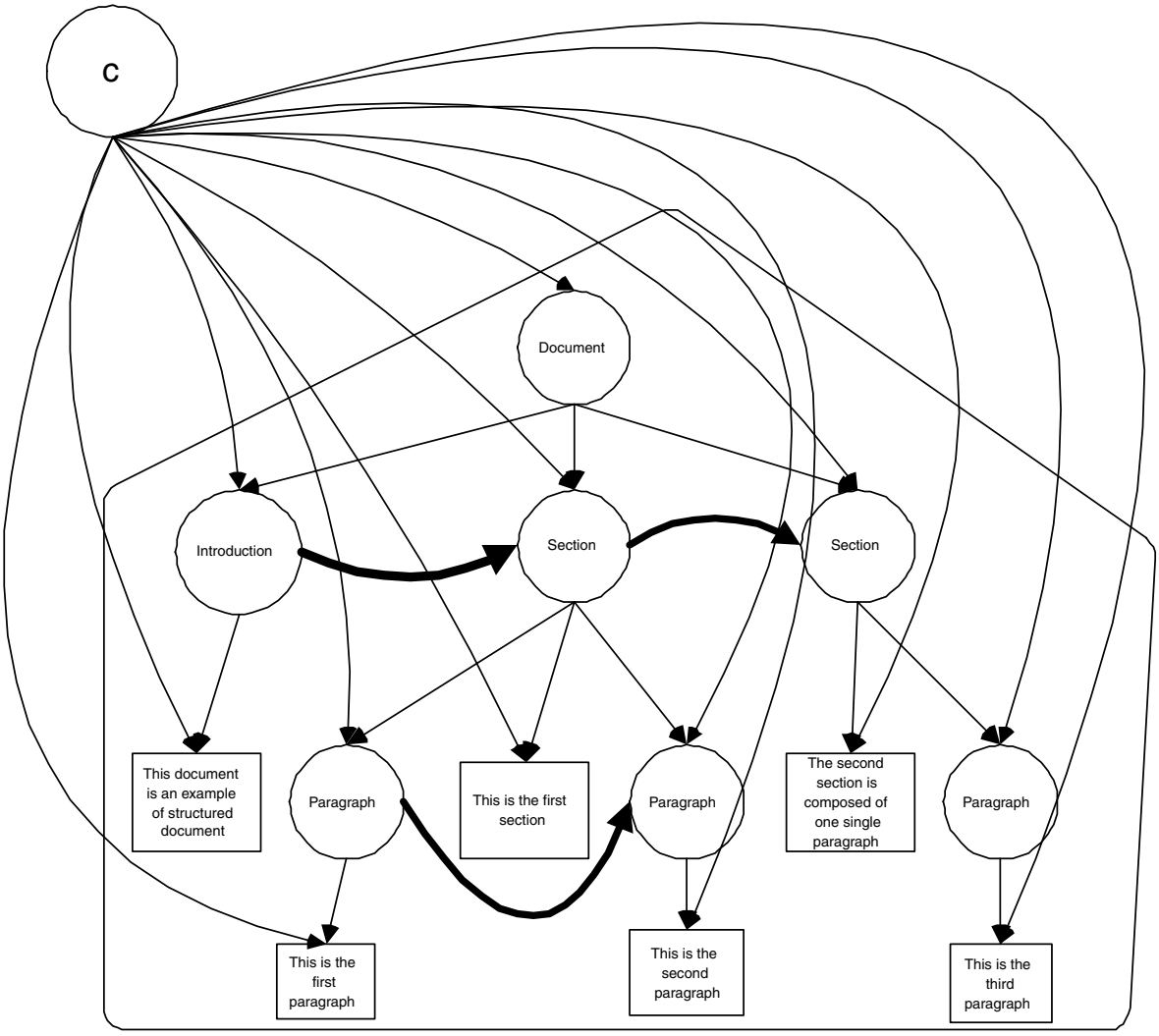


Fig. 3. The final Bayesian network making use of a TAN network at each level of the tree.

Eq. (1) can be rewritten in order to make appear a *structural* (1) and a *textual* (2) *probability*

$$P(d,c) = P(c)\underbrace{\left(\prod_{i=1}^{|d|} P(s_d^i|\mathrm{pa}(s_d^i),c)\right)}_{(1)}\underbrace{\left(\prod_{i=1}^{|d|} P(t_d^i|s_d^i,c)\right)}_{(2)} \quad (2)$$

Structural probabilities $P(s_d^i|\mathrm{pa}(s_d^i),c)$ can be directly estimated from data using some smooth estimator.

Since t_d^i is defined on the infinite set V^*, we shall make additional hypothesis for estimating the textual probabilities $P(t_d^i|s_d^i,c)$. Once again there are different possibilities for that. In the following, we use a Naive Bayes model for text fragments (Lewis, 1998), but this is not a major option and other models could do as well. Let us define t_d^i as the sequence of words $t_d^i = (w_{d,1}^i,\ldots,w_{d,|t_d^i|}^i)$ where $w_{d,k}^i \in V$ and $|t_d^i|$ is the number of

word occurrences, i.e. the length of t_d^i. According to Naive Bayes, the textual probability is rewritten as follows:

$$P(t_d^i|s_d^i, c) = \prod_{k=1}^{|t_d^i|} P(w_{d,k}^i|s_d^i, c) \tag{3}$$

The joint probability for this model is then

$$P(d, c) = P(c)\left(\prod_{i=1}^{|d|} P(s_d^i|\mathrm{pa}(s_d^i), c)\right)\left(\prod_{i=1}^{|d|}\prod_{k=1}^{|t_d^i|} P(w_{d,k}^i|s_d^i, c)\right) \tag{4}$$

Fig. 4 shows the final belief network obtained for the document in Fig. 1. For simplification, the class variable is omitted.

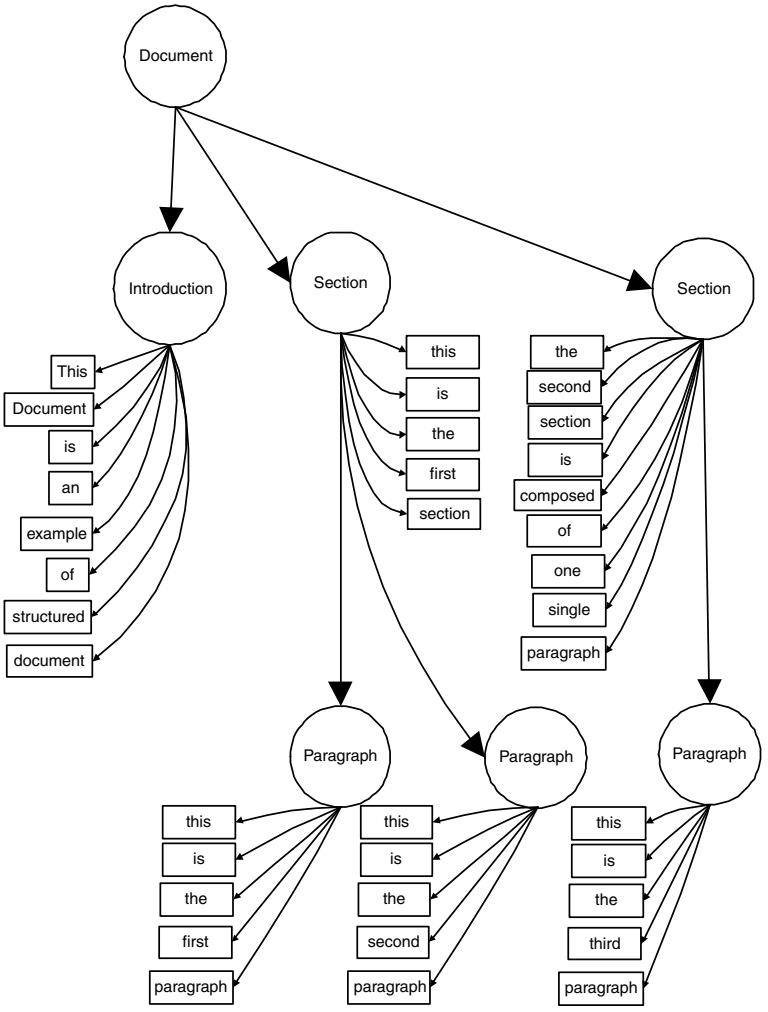


Fig. 4. The final document sub-net. In the full Bayesian network, all nodes also have node c for parent.

4.2. Classifying document parts

Eq. (4) tells us how to compute the posterior $P(c|d)$ for the whole document d. Suppose now that d is a large heterogeneous document and that fragments of d correspond to different predefined classes. We could be interested into classifying any sub-part d' of d into one of these classes. If d' corresponds to a sub-tree of d and if we consider d' out of any context, we simply use Eq. (4) by replacing d with d'. We could also be interested into classifying d' within the context of document d. For this, we need to compute $P(d', c|d_{d'})$, where $d_{d'}$ represents d with d' removed. Let s' the structural node which is the father of d' root node. Considering the structure of the Bayesian network, we get $P(d', c|d_{d'}) = P(d', c|s')$, which in turn can be estimated via:

$$P(d', c|s') = P(c)\left(\prod_{i=k'}^{|d'|+k'} P(s_d^i|\mathrm{pa}(s_d^i), c)\right)\left(\prod_{i=k'}^{|d'|+k'}\prod_{k=1}^{|t_d^i|} P(w_{d,k}^i|s_d^i, c)\right) \tag{5}$$

where k' is the index for the root of d' and structure nodes are supposed ordered according to a preorder traversal of the tree. The interesting thing here is that by computing $P(d, c)$, one automatically gets $P(d', c|d_{d'})$ and that both quantities make use of the same probabilities and probability estimates. If d' does correspond to a partial sub-tree of d instead of a full sub-tree, one gets a similar expression by limiting the structure and content terms in the products in Eq. (5) to those in d'. This is also true if d' corresponds to different sub-trees in d. Classifying d' fragments is then easily performed with the generative classifier. This compositionality property allows carrying out computations on a global object (a document) by combining the computations performed on its components (document elements). It is achieved in this model via the probabilistic conditional independence assumptions.

Compositionality is usually not a property shared by discriminant classifiers. Suppose we have a discriminant classifier trained on a vector representation of full documents. Since fragment vector representations could occupy a very different place in the vector space than full documents, the classifier will probably get very poor results when classifying these fragments. An alternative would be to train a set of classifiers on different types of document fragments. This could be prohibitive when the number of fragment types is large.

5. Learning

In order to estimate the joint probability of each document and each class, the model parameters must be learned from a training set of documents. Let us define the θ parameters as

$$\theta = \bigcup_c \left(\bigcup_{n\in\Lambda, m\in\Lambda} \theta_{n,m}^{c,s} \bigcup_{n\in V, m\in\Lambda} \theta_{n,m}^{c,w}\right) \tag{6}$$

where $\theta_{n,m}^{c,s}$ is the estimation for the $P(s_d^i = n|\mathrm{pa}(s_d^i) = m, c)$ and $\theta_{n,m}^{c,w}$ is the estimation for $P(w_{d,k}^i = n|s_d^i = m, c)$. s in $\theta_{\cdot,\cdot}^{\cdot,s}$ indicates a structural parameter and w in $\theta_{\cdot,\cdot}^{\cdot,w}$ a textual parameter. Note that for the classification task we are dealing with, there is one set of parameter for each class.

For learning the θs using the set of training documents $\mathscr{D}_{\mathrm{TRAIN}}$, we will maximize the log-likelihood L for $\mathscr{D}_{\mathrm{TRAIN}}$:

$$L = \sum_{d\in\mathscr{D}_{\mathrm{TRAIN}}} \log P(c) + \left(\sum_{i=1}^{|d|} \log \theta_{s_d^i,\mathrm{pa}(s_d^i)}^{c,s}\right) + \left(\sum_{i=1}^{|d|}\sum_{k=1}^{|t_d^i|} \log \theta_{w_{d,k}^i,s_d^i}^{c,w}\right) \tag{7}$$

The learning algorithm solves for each parameter $\theta^{c,\cdot}_{n,m}$ (where $\cdot$ corresponds to s or w) the following equation:

$$\frac{\partial L}{\partial \theta^{c,\cdot}_{n,m}} = 0 \tag{8}$$

under constraints:

$$\begin{aligned} &\forall m \in \Lambda, \quad \sum_{n \in \Lambda} \theta^{c,s}_{n,m} = 1 \\ &\forall m \in \Lambda, \quad \sum_{n \in V} \theta^{c,w}_{n,m} = 1 \end{aligned} \tag{9}$$

Let $\mathscr{D}^c_{\text{TRAIN}}$ be the sub-set of all documents in $\mathscr{D}_{\text{TRAIN}}$ with class c, let $N^{d,c}_{n,m}$ be the number of times a node with label n (in V or Λ) has his parent with label m in document d, the solution of the learning problem is then

$$\theta^{c,\cdot}_{n,m} = \frac{\sum_{d \in \mathscr{D}^c_{\text{TRAIN}}} N^{d,c}_{n,m}}{\sum_i \sum_{d \in \mathscr{D}^c_{\text{TRAIN}}} N^{d,c}_{i,m}} \tag{10}$$

For the experiments we will use the following smooth estimator:

$$\theta^{c,\cdot}_{n,m} = \frac{\sum_{d \in \mathscr{D}^c_{\text{TRAIN}}} N^{d,c}_{n,m} + 1}{\sum_i \sum_{d \in \mathscr{D}^c_{\text{TRAIN}}} N^{d,c}_{i,m} + \Delta} \tag{11}$$

where Δ is equal to $|V|$ if $n \in V$ or $|\Lambda|$ if $n \in \Lambda$.

The complexity of this learning algorithm is $O(\sum_{\mathscr{D}^c_{\text{TRAIN}}} |d| + |t_d|)$. In a classical structured document, the number of structural nodes $|d|$ is usually smaller than the number of words $|t_d|$. The complexity is then $O(\sum_{\mathscr{D}^c_{\text{TRAIN}}} |t_d|)$ which is the complexity of the classical Naive Bayes model on flat documents (Lewis, 1998).

The generative classifier can cope with both the content and structure information of structured documents. It also allows one to perform inference on the different nodes and sub-trees of the network. Document parts can then be classified in the context of the whole document. More generally decisions can be made by taking into account only a sub-part of the document or when information is missing in the document.

We will show below that this classifier can also be turned into a discriminant classifier. Discriminant techniques are known to be generally more efficient for classification than generative ones. Note that this is not the only criterion for selecting a classifier. For example doing so, we lose the possibility to easily add new classes. As discussed in Section 4.2, inference on document parts is also much less natural with this approach.

6. Improving discriminant abilities using Fisher kernel

In order to improve the discriminative abilities of generative models, Jaakkola, Diekhans, and Haussler (1999) proposed a new method based on the Fisher scores. They developed this method for classifying sequences modeled with HMMs and this led to a significant performance increase on biological data. We show below how this idea naturally extends to tree generative models.

6.1. Fisher score and Fisher kernel

The key idea of Jaakkola et al. (1999) is to derive from a learned generative model, a representative vector for each example x and to use a classical discriminant classifier on this vector representation.

The fisher score for an example x and a generative model with parameters θ is defined as

$$U_x = \nabla_\theta \log P(x|\theta) \tag{12}$$

where ∇_θ is the gradient operator with respect to the θ parameters.

The Fisher score is a fixed-size vector which explains how much parameters of the model do contribute to generate the example. Using this vector, we can then create a kernel function $K(x,y)$ as follows:

$$K(x,y) = U_x^{\mathrm{T}} M^{-1} U_y \quad \text{with } M = E_X[U_X^{\mathrm{T}} U_X] \tag{13}$$

This kernel function can be used with any kernel-based classifier like for example support vector machines. This kernel defines a distance between two examples x and y. We explain below how the Fisher kernel could be used with our model.

6.2. Fisher score for the Bayesian network model

We shall consider one Fisher score for each document d and each class c defined as

$$U_d^c = \nabla_{\theta^{c,\cdot}_{\cdot,\cdot}} P(d,c|\theta) \tag{14}$$

Using Eq. (7), we compute the Fisher score component for each parameter $\theta^{c,\cdot}_{n,m}$:

$$\frac{\partial \log P(d,c|\theta)}{\partial \theta^{c,\cdot}_{n,m}} = \frac{N^{d,c}_{n,m}}{\theta^{c,\cdot}_{n,m}} \tag{15}$$

6.3. Computational considerations

Additional tricks are needed to make the Fisher kernel method work. We use two such modifications here. They have been chosen after different trials. The first simplification is to approximate the M matrix with the identity matrix as it is done in Hofmann (2000), Vinokourov and Girolami (2001) and Jaakkola et al. (1999). In order to get a better scaling of the vector features, we also compute the gradient of the likelihood with respect to $2\sqrt{\theta^{c,\cdot}_{n,m}}$ instead of $\theta^{c,\cdot}_{n,m}$, this has been suggested in Hofmann (2000). The final formula for our model is then

$$\frac{\partial \log P(d,c|\theta)}{\partial 2\sqrt{\theta^{c,\cdot}_{n,m}}} = \frac{N^{d}_{n,m}}{\sqrt{\theta^{c,\cdot}_{n,m}}} \tag{16}$$

Let $\Lambda = \{\lambda_i\}_{i\in[1,\ldots,|\Lambda|]}$ and $V = \{v_i\}_{i\in[1,\ldots,|V|]}$; for each document d and class c we obtain a vector $v^{c,d}$ corresponding to the document representation in the Fisher space. This vector is detailed in Fig. 5.

We used here the kernel function with a binary SVM classifier. For each class c, we used the kernel function corresponding to the generative model learned on this class. The Fisher score is proportional to the frequency of each word in a particular tag and decreases when $\theta^{c,\cdot}_{n,m}$ increases. tf-idf behaves almost similarly: it is an increasing function of the term frequency and it decreases when the word appears in many documents. The main difference is that df is computed on the whole collection whereas $\theta^{c,\cdot}_{n,m}$ depends on class c.

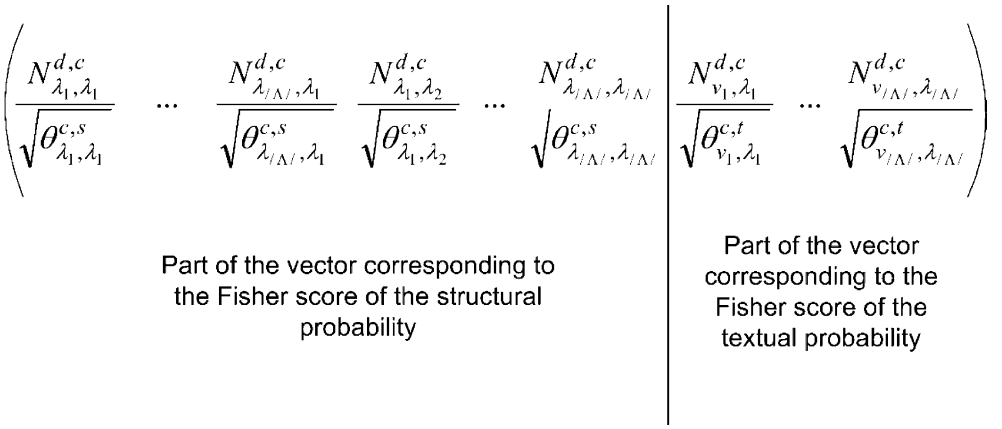


Fig. 5. The Fisher score for document d in class c. This vector is composed of two parts: one for the structural probability and another for the textual probability.

7. Considering different types of information: text and image

The above models could be used not only with text, but with any other type of content. The only requirement is that we have a generative model for scoring the different content types. We describe below an extension of the structured document model using a generative model for images. It will be used in Section 8.3 for classifying Web pages using both text and image information. We do not ambition here to describe a state of the art model for image classification. Instead, we merely want to show, using as an example a basic image model, how different modalities could be handled in the structured document classifier. More sophisticated image models could be used in order to get better performance.

Let us now consider that t_d^i can be either text or image. The new set of labels will be denoted by $\Lambda_2 = \Lambda \cup \{I\}$ where as before Λ is the set of label for textual content and I is a unique label for nodes with image information (multiple image labels could be considered as well).

For the remainder of the section, t_d^i represents either text information if $s_d^i \in \Lambda$ or image information if $s_d^i = I$.

Eq. (4) can be rewritten:

$$\begin{aligned} P(d,c) &= P(c)\left(\prod_{i=1}^{|d|} P(s_d^i|\mathrm{pa}(s_d^i),c)\right)\left(\prod_{i=1}^{|d|} P(t_d^i|s_d^i,c)\right) \\ &= P(c)\left(\prod_{i=1}^{|d|} P(s_d^i|\mathrm{pa}(s_d^i),c)\right)\left(\prod_{i=1/s_d^i\in\Lambda}^{|d|} P_{\text{text}}(t_d^i|s_d^i,c)\right)\left(\prod_{i=1/s_d^i=I}^{|d|} P_{\text{image}}(t_d^i|s_d^i,c)\right) \end{aligned} \tag{17}$$

where $P_{\text{text}}(t_d^i|s_d^i,c)$ is a textual probability and $P_{\text{image}}(t_d^i|s_d^i,c)$ is an image probability.

Our image model is the following: we rescale all images to a fixed size of N_p pixels and represent this scaled image by an histogram consisting of N_{color} colors. We then use the following Naive Bayes model:

$$P_{\text{image}}(t_d^i|s_d^i,c) = \prod_{k=1}^{N_{\text{color}}} P(p_{d,k}^i|s_d^i,c), \quad \forall \sum_{k=1}^{N_{\text{color}}} p_{d,k}^i = N_p \tag{18}$$

where $P(p_{d,k}^i|s_d^i,c)$ is the probability that the image of node i has $p_{d,k}^i$ pixels of color k.

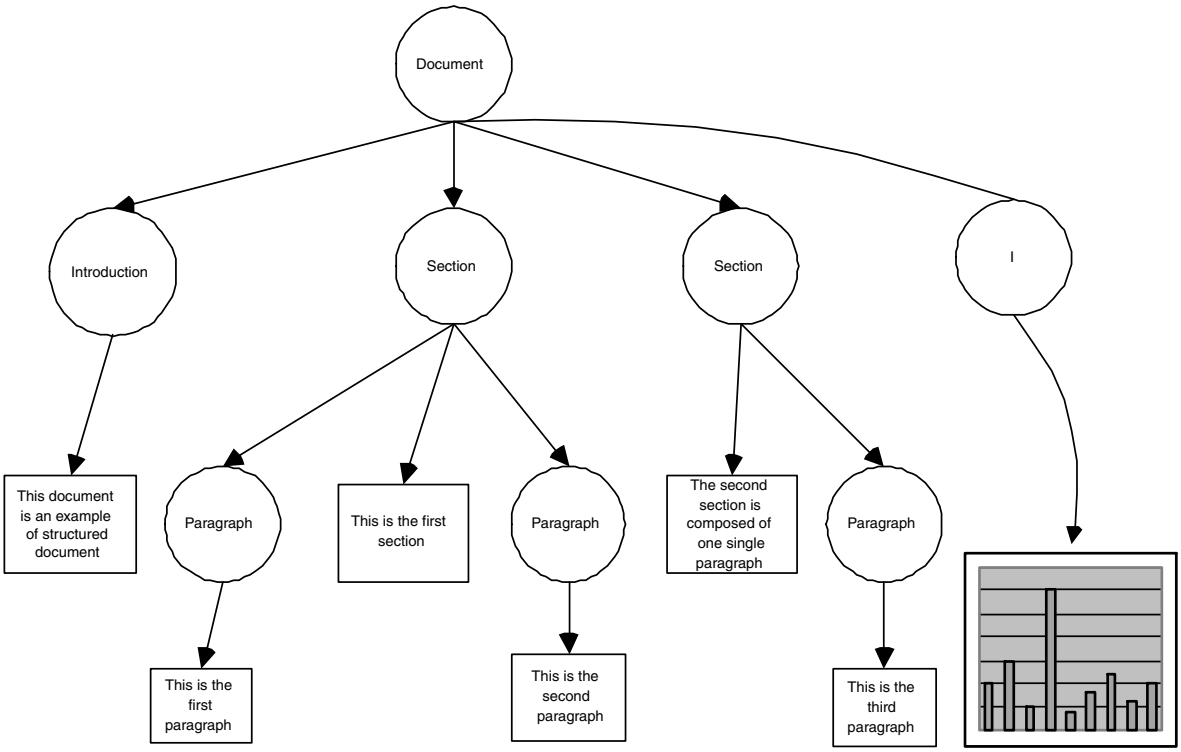


Fig. 6. A text + image document: the image is represented by an histogram in the lower right corner.

Although this is a basic image model, it performed reasonably well for our tests. We also did some experiments by adding texture and form characteristics to the color characteristic. This did not led to any improvement compared to color alone. Similar observations have also been made by many authors for Web images classification.

Figs. 6 and 7 represent respectively a multimedia document and the associated Bayesian network.

Eq. (17) can now be rewritten:

$$P(d,c) = P(c)\left(\prod_{i=1}^{|d|} P(s_d^i|\mathrm{pa}(s_d^i),c)\right)\left(\prod_{i=1/s_d^i\in\Lambda}^{|d|}\prod_{k=1}^{|t_d^i|} P(w_{d,k}^i|s_d^i,c)\right)\left(\prod_{i=1/s_d^i=I}^{|d|}\prod_{k=1}^{N_{\mathrm{color}}} P(p_{d,k}^i|s_d^i,c)\right) \tag{19}$$

8. Experiments

The structured document model has been tested on three different corpora. We performed extensive experiments on the INEX corpus (Fuhr et al., 2002) which is a large collection of XML documents. We present additional experiments on a corpus of textual HTML pages (webKB, 1999) which has already been used by different authors for comparing flat classifiers. Finally we show how the model behaves on multimedia data. The corpus for this experiment has been gathered inside the European project NetProtect for testing Web page filtering tools. Since the task of classifying XML documents is new, there is not yet any relevant corpus for evaluating document fragment classification. For the three corpora, we then present

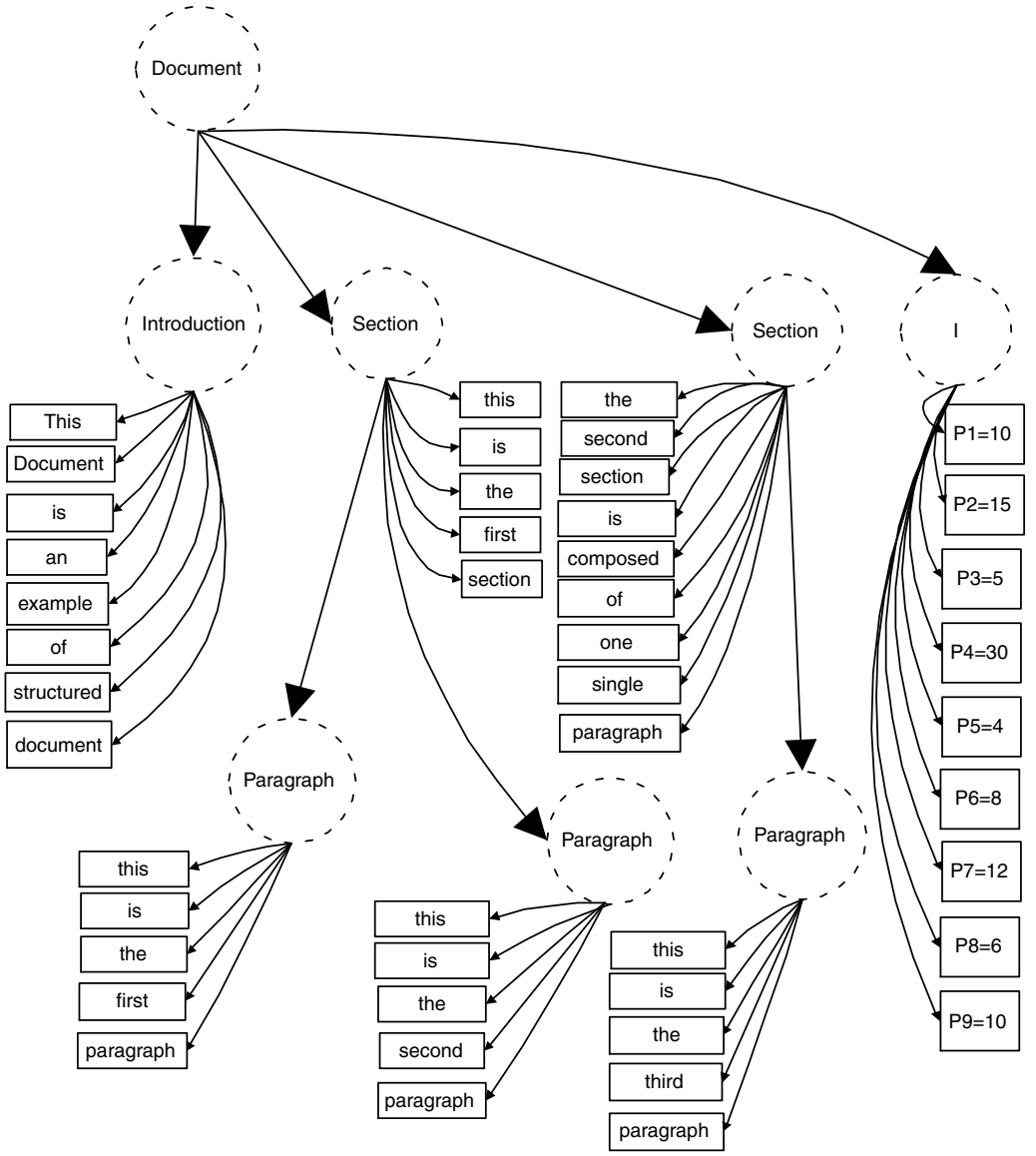


Fig. 7. The network built for the document in Fig. 6. $Pk = u$ for an image node means that there are u pixel with color k in the image.

tests only for the classification of whole documents. We first present below the results obtained on INEX and then discuss experiments on webKB and NetProtect.

8.1. INEX: a large XML corpus

The INEX corpus has been recently developed for ad hoc retrieval on XML documents. It is composed of 15,000 articles from journals and proceedings of the IEEE Computer Society. Articles are issued from 18 different journals or proceedings. There are about seven million nodes (doxels) and each document is a tree

of about 590 nodes. This corpus was not initially aimed at categorization, however it is reasonable to believe that the different journals and proceedings may be classified according to their content and structure. We thus chose as classes the 18 different journal and proceeding "headers". Documents were preprocessed using the Porter stemmer and words appearing in less than 20 documents were pruned. In order to keep only the body of the articles, the header of each document which explicitly contains the class of the document was also removed. Note that each document is assigned to a unique class.

8.1.1. Evaluation measure

We performed two classification tasks on the INEX corpus. The first one is multi-class single label classification (see Sebastiani, 2002) where each document is assigned a unique class. The second one is document ranking. The goal here is to rank the documents according to their relevance for each class. This is the usual binary classification problem considered in IR. Performance measures are different for each task and are detailed below.

8.1.1.1. Multi-class single label categorization. This is the classical classification task in machine learning. The usual measure is the classification accuracy. The set of classes is $C = \{c_i\}_{i\in[1,\ldots,|C|]}$.

The class c_d of a document d is predicted according to

$$c_d = \underset{c\in C}{\operatorname{argmax}} P(c|d) \tag{20}$$

We have used as measure the macro and micro-average accuracies. The former is the mean accuracy over all classes and the latter is the mean accuracy when each class is weighted by its size. Macro-average puts an equal weight on all classes whereas micro-average is dominated by large classes. Note that since each document is assigned a unique class, accuracy is equivalent to micro-recall. Since we are only interested into classifying documents in one class, precision does not bring any additional information and it is omitted here.

8.1.1.2. Ranking. For each class, the documents are ranked according to their probability $P(c|d)$. The classical evaluation measure here is the precision–recall curve. As in Joachims (1998), we use the micro-average break-even point and the macro-average break-even point to summarize these curves.

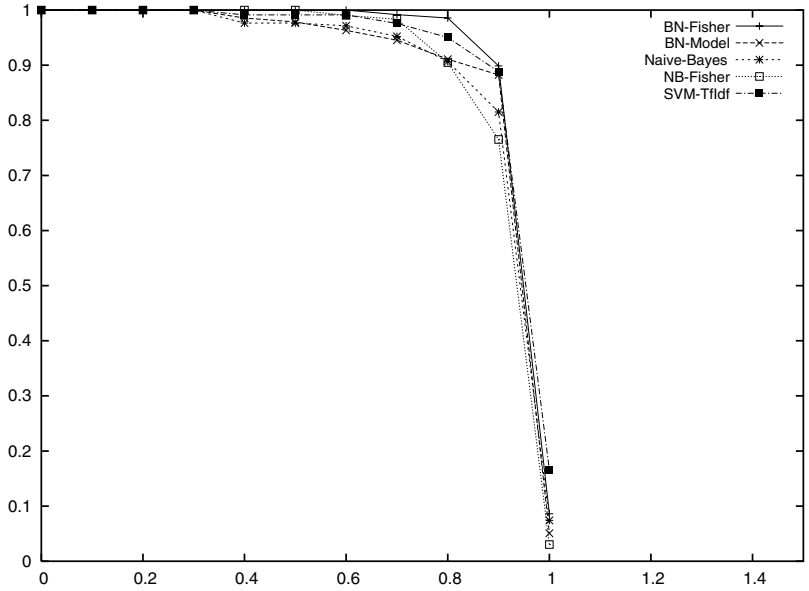
8.1.1.3. Results. We used two baseline models for comparison, Naive Bayes and SVM. SVM has been used with a tf-idf encoding (Joachims, 1998). We have used the binary classifier version of SVM where one class is learned against all others. SVM outputs have been scaled to $[0, 1]$ using the algorithm proposed by Platt (2000). In the structured generative classifier, large elements tend to dominate the model score. This is not desirable since elements with only a few words may be characteristic of a given class. We then normalized the scores of the textual document elements for giving a similar importance to elements with different length. The Fisher method was evaluated with both Naive Bayes (*Naive-Bayes Fisher*) and the structured document classifier (*Bayesian network Model Fisher*).

Multi-class single label results are given in Fig. 8. Figs. 9 and 10 respectively show the micro- and macro-average precision–recall curves obtained for document ranking.

If we compare the generative models, for both multi-class categorization and ranking, the Bayesian network model outperforms Naive Bayes. For multi-class categorization, the improvement is about 6% on macro-average recall and 2% on micro-average recall which means that improvements mainly concern small classes. For ranking, the improvement is about 4% on macro-average breakeven point and 5% on micro-average. The two precision–recall curves in Figs. 9 and 10 show a clear improvement of the ranking quality using the structured generative model over Naive Bayes. As could be expected, all discriminant models do outperform the generative ones on the multi-class single label document classification task. The

	Macro-average recall	Micro-average recall
Naive Bayes	61 [59.8; 62.1]	64 [62.8; 65.1]
BN Model	67 [65.8; 68.1]	66 [64.8; 67.1]
SVM	71 [69.8; 72]	70 [68.8; 71.1]
Naive Bayes Fisher	69 [67.8; 70.1]	69 [67.8; 70.1]
BN Model Fisher	72 [70.8; 73]	71 [69.8; 72]

Fig. 8. Performance of the textual models over the INEX corpus: macro-average and micro-average recall with 95% confidence intervals in multi-class single label classification.

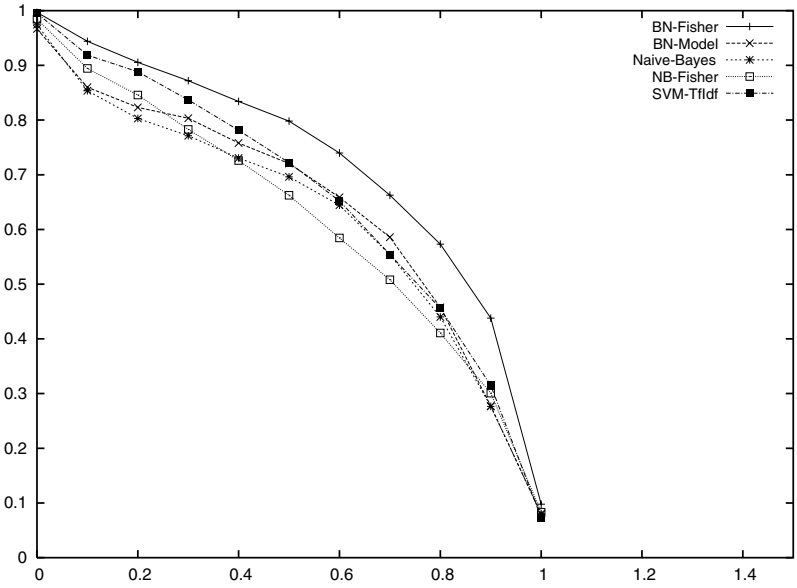


	BreakEven Point
Naive Bayes	0.85
Bayesian Model	0.9
SVM TF-IDF	0.91
Naive Bayes Fisher	0.83
BN Fisher	0.92

Fig. 9. Performance over the INEX corpus: micro-average precision–recall curves and micro-average breakeven point for the ranking task.

Fisher model for the Bayesian network is slightly better than SVM, but this is not significant since the confidence intervals overlap. For ranking, micro-average performance of all models are very close, this is because one of the large classes dominates. The macro-average precision–recall curves show that the Fisher Bayesian network model outperforms all others and that the generative Bayesian network is very close to SVM.

Note that this is the first time a classification experiment has been performed on this recent and reference XML collection.



	BreakEven point
Naive Bayes	0.58
Bayesian Model	0.62
SVM TF-IDF	0.62
Naive Bayes Fisher	0.62
BN Fisher	0.68

Fig. 10. Performance over the INEX corpus: macro-average precision–recall curves and macro-average breakeven point for the ranking task.

8.2. HTML corpus: webKB

The webKB corpus (webKB, 1999) has become a reference corpus in the machine learning community for HTML page categorization. It is composed of 8282 HTML documents from computer science department Web sites. These documents are issued from seven topics, but we used only six topics (*student*, *faculty*, *course*, *project*, *department* and *staff*) as is usually done. We are then left with 4520 documents. For preprocessing, we used Porter Stemming and pruned words that appeared in less than five documents. The vocabulary V is then composed of 8038 terms. We only kept tags with higher frequency (*H1*, *H2*, *H3*, *TITLE*, *B*, *I*, *A*). We used a fivefold cross-validation with 80% of the documents for training and 20% for testing.

Experiments on this HTML corpus confirm the results obtained on INEX which shows that the model is valid for different types of structured documents. We then present results only for multi-class single label categorization, using the micro-average and the macro-average accuracy as is done in Diligenti et al. (2001). The results are given in Fig. 11.

Macro-average accuracy is similar respectively for the two generative models and for the three discriminant models. The latter models outperform the former ones. With respect to micro-average, the struc-

	Macro-average recall	Micro-average recall
Naive Bayes	70 [68.4;71.4]	81 [79.6;82.2]
BN Model	70 [68.4;71.4]	83 [81.7;84.1]
SVM	73 [71.5;74.4]	85 [83.7;86.1]
Naive Bayes Fisher	72 [70.5;73.4]	85 [83.7;86.1]
BN Model Fisher	73 [71.5;74.4]	87 [85.8;88]

Fig. 11. Performance on the webKB corpus: macro-average and micro-average recall with 95% confidence intervals in multi-class single label classification.

	Macro-average recall	Micro-average recall
Naive Bayes	89.9 [89.2;90.4]	88.4 [87.7;89]
BN Text only	92.5 [91.9;93]	92.9 [92.3;93.3]
BN Image only	83 [82.2;83.7]	82.7 [81.9;83.4]
BN Text+Image	93.6 [93.1;94]	94.7 [94.2;95.1]

Fig. 12. Performance of text + image model over the NetProtect corpus: macro-average and micro-average recall with 95% confidence intervals.

tured generative model is 2% better than Naive Bayes and the Fisher Bayesian network is also 2% better than the tf-idf SVM model. Confidence intervals overlap in both cases.

8.3. Multimedia corpus: NetProtect corpus

The text and image model was tested over a corpus of 19,652 HTML pages with two topics (classes). All pages in this corpus contain text and image informations. We pruned each word that appeared in less than 50 documents. We then used 50% of the documents for training and the remaining 50% for testing. This experiment was designed for testing the ability of the structured generative model to handle different information types. We thus only performed experiments with the generative models here. The Bayesian network was also tested using the text and image page content separately. The results are shown in Fig. 12.

They show that the Structured model significantly outperforms Naive Bayes (+6% for micro-average and +4% for macro-average) and is able to take benefit from the two information sources present in the page.

Globally, the proposed generative model performed well on the three different collections. For full document classification, its performance is above the baseline Naive Bayes while being still less than that of discriminant classifiers. For the ranking task on the INEX corpus, the model performance is similar to SVM. This generative model has desirable capabilities: it is easy to integrate different information sources and to perform inference on document parts. These properties are usually not shared by discriminant methods operating on vector representations. We believe that these qualities are essential for future applications of structured document classifiers. The Fisher model built on top of the generative one is a discriminant vector classifier which slightly improves the baseline SVM.

9. Conclusion

We have presented a new generative model for structured document. It is based on Bayesian networks and allows one to model the structure and the content of documents. It has been tested here for the classical

task of whole document classification. We have described how this model can be turned into a discriminant model using the Fisher kernel method. We have also shown how our model can be easily extended to take into account different types of information and have presented an example of multimedia classification. We have performed tests on three different document collections which show that the model behaves well on a variety of situations. Further investigations are needed for analyzing its behavior on document fragments classification. The model could also be modified for learning implicit relations between document elements besides using the explicit structure. An interesting aspect of the generative model is that it could be used for other tasks relevant to IR. It could serve as a basis for clustering structured documents. The difference with the classification application is that in this case we miss the class information. The natural solution is to consider a mixture of Bayesian network models where parameters do depend on the mixture component instead of the class as it is the case here. Estimation maximization (EM) equations can be easily derived for such a mixture model. Schema mapping and document structuring are new tasks that are currently being investigated in the database and IR communities. The potential of the model for performing inference on document parts when information is missing in the document could be helpful for this type of application.

References

Baeza-Yates, R., Carmel, D., Maarek, Y., & Soffer, A. (Eds.). (2002). *Journal of the American Society for Information Science and Technology (JASIST)*.

Blei, D. M., & Jordan, M. I. (2003). Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR* (pp. 127–134). ACM Press.

Cai, L., & Hofmann, T. (2003). Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th annual international ACM SIGIR* (pp. 182–189). ACM Press.

Callan, J. P., Croft, W. B., & Harding, S. M. (1992). The INQUERY retrieval system. In *Proceedings of DEXA-92* (pp. 78–83).

Chakrabarti, S., Dom, B. E., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In L. M. Haas & A. Tiwary (Eds.), *Proceedings of SIGMOD-98, ACM international conference on management of data* (pp. 307–318). New York, Seattle, USA: ACM Press.

Cline, M. (1999). *Utilizing HTML structure and linked pages to improve learning for text categorization*. Undergraduate Honors Thesis, Department of Computer Science, University of Texas.

Denoyer, L., Zaragoza, H., & Gallinari, P. (2001). HMM-based passage models for document classification and ranking. In *Proceedings of ECIR-01* (pp. 126–135). Darmstadt, DE.

Diligenti, M., Gori, M., Maggini, M., & Scarselli, F. (2001). Classification of HTML documents by hidden tree-Markov models. In *Proceedings of ICDAR* (pp. 849–853). Seattle, WA, USA.

Dumais, S. T., & Chen, H. (2000). Hierarchical classification of Web content. In N. J. Belkin, P. Ingwersen, & M.-K. Leong (Eds.), *Proceedings of SIGIR-00* (pp. 256–263). ACM Press.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: analysis and applications. *Machine Learning, 32*(1), 41–62. Available: http://citeseer.nj.nec.com/fine98hierarchical.html.

Fuhr, N., Govert, N., Kazai, G., & Lalmas, M. (2002). INEX: Initiative for the evaluation of XML retrieval. In *Proceedings ACM SIGIR 2002 workshop on XML and information retrieval*. Available: http://inex.is.informatik.uni-duisburg.de:2003.

Hofmann, T. (1999a). The cluster-abstraction model: unsupervised learning of topic hierarchies from text data. In *IJCAI* (pp. 682–687).

Hofmann, T. (1999b). Probabilistic latent semantic analysis. In *Proceedings of uncertainty in artificial intelligence, UAI'99*. Stockholm.

Hofmann, T. (2000). Learning the similarity of documents: an information-geometric approach to document retrieval and categorization. In *Research and development in information retrieval* (pp. 369–371).

Jaakkola, T. S., Diekhans, M., & Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. In *Intelligent systems for molecular biology conference (ISMB'99)*. Heidelberg, Germany: AAAI.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98* (pp. 137–142). Heidelberg, DE, Chemnitz, DE: Springer Verlag.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In D. H. Fisher (Ed.), *Proceedings of ICML-97, 14th international conference on machine learning* (pp. 170–178). San Francisco, Nashville, USA: Morgan Kaufmann Publishers.

Lewis, D. D. (1998). Naive (Bayes) at forty: the independence assumption in information retrieval. In *Proceedings of ECML-98* (pp. 4–15). Heidelberg, DE, Chemnitz, DE: Springer Verlag. Available: http://www.research.att.com/lewis/papers/lewis98b.ps.

Myaeng, S. H., Jang, D.-H., Kim, M.-S., & Zhoo, Z.-C. (1998). A flexible model for retrieval of SGML documents. In *Proceedings of the 21st annual international ACM SIGIR* (pp. 138–140). New York, Melbourne, Australia: ACM Press.

Piwowarski, B., Faure, G., & Gallinari, P. (2002). Bayesian networks and INEX. In *Proceedings of the first annual workshop of the initiative for the evaluation of XML retrieval (INEX). DELOS workshop*. Dagstuhl, Germany: ERCIM.

Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schoelkopf, & D. Schuurmans (Eds.), *Advances in large margin classifiers* (pp. 61–74).

Quek, C. Y. (1997). *Classification of World Wide Web documents*. Senior Honor Thesis, School of Computer Science, Carnegie Mellon University.

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

Termier, A., Rousset, M., & Sebag, M. (2002). Treefinder: a first step towards XML data mining. In *ICDM* (pp. 450–457).

Vinokourov, A., & Girolami, M. (2001). Document classification employing the Fisher kernel derived from probabilistic hierarchic corpus representations. In *Proceedings of ECIR-01* (pp. 24–40). Darmstadt, DE.

webKB (1999). Available: http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.

Yang, Y., Slattery, S., & Ghani, R. (2002). A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems, 18*(2/3), 219–241.

Yi, J., & Sundaresan, N. (2000). A classifier for semi-structured documents. In *Proceedings of the sixth ACM SIGKDD* (pp. 340–344). ACM Press.

Zaki, M. J., & Aggarwal, C. C. (2003). Xrules: An effective structural classifier for xml data. In *SIGKDD 03*. Washington, DC.