Journal of Universal Computer Science, vol. 14, no. 18 (2008), 2998-3010
submitted: 1/3/08, accepted: 29/9/08, appeared: 1/10/08 © *J.UCS*

Stacked Dependency Networks for Layout Document Structuring

Boris Chidlovskii
(Xerox Research Centre Europe, Meylan, France
chidlovskii@xrce.xerox.com)

Loïc Lecerf
(Xerox Research Centre Europe, Meylan, France
lecerf@xrce.xerox.com)

Abstract: We address the problems of structuring and annotation of layout-oriented documents. We model the annotation problems as the collective classification on graph-like structures with typed instances and links that capture the domain-specific knowledge. We use the relational dependency networks (RDNs) for the collective inference on the multi-typed graphs. We then describe a variant of RDNs where a stacked approximation replaces the Gibbs sampling in order to accelerate the inference. We report results of evaluation tests for both the Gibbs sampling and stacking inference on two document structuring examples.

Key Words: Document structuring, dependency networks, stacking
Category: H.2, H.3.7, H.5.4

1 Introduction

Large collections of layout-oriented documents are collected and stored in digital libraries and content management systems. The documents, often available in electronic formats (PS, PDFs, MS Word, etc.), originate from domain-specific collections or get harvested over the Web. To organize an access to documents, the systems use the descriptive metadata given by physical and bibliographic attributes (url, media, size, title, author, keywords, etc.). Moreover, the systems proceed to a further annotation and fine-grained *document structuring* in order to facilitate the internal navigation, information extraction and visualization [Chanod et al.(2005)]. Giving the internal structure to the layout documents is achieved by 1) segmenting them into *logical* fragments including pages, paragraphs, sections, etc., and 2) adding *semantic* annotations, creating the table of contents, linking elements to indexes and topic taxonomies.

Existing logical analysis algorithms [Mao et al.(2003)] do allow to accomplish some of these tasks; however, their accuracy may suffer in the frequent cases of document heterogeneity. To increase the robustness of document structuring methods, the current systems attempt to complete the logical analysis with methods based on the learning and classification.

Conventionally, the *classification techniques* start with a segmentation of documents into lines, blocks and zones accomplished by the document logical analysis. The classifiers build the prediction models for the document elements by associating an element label to some element's characteristics, such as its x-y positions on the page, the font size or the presence of certain keywords in the textual information. Beyond the proper element characteristics, the layout-oriented documents may hide important meaningful relationships between elements. For example, labeling a line as a bibliographic reference becomes more certain if the previous and next lines are already annotated as references. Obviously, capturing such relationships and their integration in classifiers can help train more accurate predictive models for the document annotation.

The previous classification methods have been primerily dedicated to the one-page structuring tasks. Techniques based on statistical grammars have been used; in particular, Hidden Markov Machines (HMMs) [Kopec and Chou(1994)] and Probabilistic Context-Free Grammars (PCFGs) [Liang et al.(2005)] have been extended for the 2-dimensional case in order to model and decode the page structure. However, moving to the *multi-page documents* can not be achieved by mechanically extending the previous methods. Indeed, the grammar-based techniques show certain limitations in the modeling and scalability issues. On one side, HMMs work with the sequence-like data representation only and can not model long-distance relationships. On the other size, PCFGs do model long-distance relationships; however the $O(N^3)$ inference complexity, where N is the number of elements, is hardly scalable to documents with hundrends of pages.

In this paper we address the large scale document structuring and annotation problems and investigate techniques alternative to the grammar-based ones. We model a document annotation task as a collective annotation of groups of related instances rather than the individual label annotation for each instance. The user is invoked to define a set of link features to order to capture the domain- or task-specific knowledge present in documents. The full set of elements and user-defined links infer a *relational dependency network* (RDN) [Neville and Jensen(2007)] where both elements and links can be *typed*. Moreover, both elements and links may be annotated. In general, we attempt to reiterate the success the dependency networks have had in different domains, including the social networking and the Web pages classification and extend them to the layout document structuring.

Relational dependency networks (RDNs) is a graphical model that is capable to express inter-element dependencies and to reason in the relational setting. The RDNs decompose the globally trained model into a set of local learning models, where local features get extended with the estimations of labels of relevant elements. First, the conventional Gibbs sampling can be used for the collective inference. When the Gibbs sampling shows a slow convergence, we follow [Kou and Cohen(2007)] in replacing the Gibbs sampling with the stacked assembly principle that allows to accelerate the inference.

The remainder of the paper is organized as follows. In Section 2 we explain the problem of annotating the layout-oriented documents and introduce the mapping from input data to the set of output variables. Section 3 describes the relational dependency networks and the inference with the Gibbs sampling. It then introduces the stacking meta-learning principle and proposes the learning and inference algorithms. Finally, Section 4 reports evaluation tests on two collections and Section 5 concludes the paper.

2 Document annotation modeling

This work is motivated by a number of large-scale document structuring and annotation problems. One example is the metadata extraction from layout-oriented documents, where documents expose different page and markup templates, like scientific papers, technical manuals, etc. The extraction process concerns such metadata attributes as titles, authors, organizations, addresses and references. Another example is the classification and semantic annotation of Web pages available in HTML format.

In the following the input data is given by a structure describing a set of elements in document(s) and relationships among them. In the case of Web pages, this structure is given by HTML tag nesting and hyperlinks. In the scanned documents, the OCR and the logical analysis algorithms are deployed on the preliminary step in order to identify characters, words, lines or zones. They produce an hierarchical structure as the spatial nesting ("*contains*" relationship) of document fragments and include elements like `LINE`s, `PARAGRAPH`s, `TABLE`s, etc. In certain document applications, like forms or business cards recognition, the 2-dimensional page analysis can go further and establish spatial logical relationships between the elements, like *"touch"*, *"intersect"*, etc.

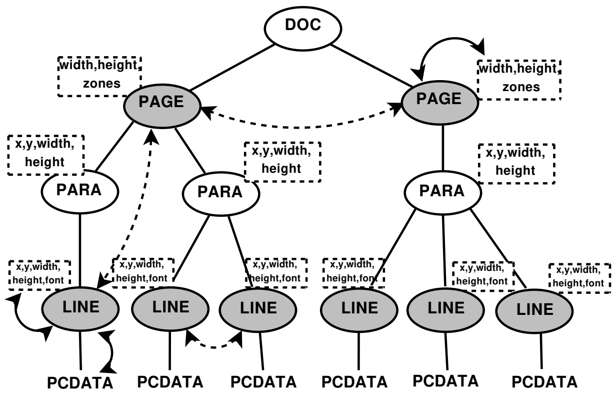

Figure 1: XML structure of input documents.

Beyond the basic line and zone segmentation and the spatial containment, an advanced logical analysis can establish links (or link candidates) for a specific logical composition of a physical document. For example, consider the Table of Content (ToC) detection in multi-page documents [Dejean and Meunier(2005)]. Under some generic constraints of the contiguity and textual similarity, the ToC detection algorithm can produce the candidate pairs for being linked to the ToC. Similar examples are linking to bibliographic references and citations.

Given the input data structure, the classification techniques try to build the prediction models for the document elements. They proceed by extracting *local features* which associate an element label to element's characteristics, such as elements' x-y positions on the page, the font size or certain keywords in the textual data.

Beyond the local features, so called *link features* try to capture relationships between elements' labels, for example, between neighbor elements in the document. Most layout-oriented documents are full of meaningful relationships. For example, labeling a line as a bibliographic reference in a scientific paper becomes more certain if the previous and next lines are already recognized as the references. Capturing such relationships and their integration in the classifiers can help train more accurate predictive models for the document annotation.

Any link feature is defined over a pair of elements or fragments in input documents. The link can either match the existing relationships or form a new one. The full set of elements and links induce a graph-like structure. The collective assignment of element labels are described by joint probability distributions in *probabilistic graphical models*, in particular Markov random fields [Bishop(2006)]. If the graph forms a chain or a tree, HMM or Conditional Random Fields (CRF) techniques can be used to find an optimal joint assignment of labels by optimizing a log-likelihood function [Lafferty et al.(2001)]. If the structure of the relationships between elements form an undirected graph, finding exact solutions require special graph transformations [Bishop(2006)] and eventually the enumeration of all possible annotations on the graph. This results in the exponential complexity of model training and inference. To make it tractable, several approximation techniques have been proposed for undirected graphs; these include Markov Chain Monte Carlo methods and dependency networks [Neville and Jensen(2007)].

In layout-oriented documents, the collective annotation targets the groups of document elements and fragments. Figure 2 shows another document structuring example. Given a sequence of pages (three sequential pages are shown in the figure, where the textual data and the paragraphs are omitted for the sake of presentation), the task consists of segmenting the document in sections. It should detect the starting page of each section and its title. The corresponding annotations are marked by the bounding box for the second page and two black lines that compose the section title.

2.1 Mapping

For the document segmentation in Figure 2, the basic classification approach would treat line and pages separately, one by one. In the collective classification, dependencies between elements can give more evidences for making some annotations and thus

Figure 2: Example of the collective element annotation.

improve the accuracy of the prediction models.

The shift from the separate classification to collective one is enabled through the mapping from input data to output variables and the dependencies. In the following, we will slightly change the standard RDN notation [Neville and Jensen(2007)], in order to explicitly express this mapping. We define a 2-layered graph $G = \{In, Out, Map\}$, where the lower layer In describes the input elements and relationships between them, and the upper layer Out describes output variables and dependencies between them. The mapping description Map first defines the variable sets and associates them to input elements. Second, it defines templates for both local and link features on input data and output variables. Finally, the modeling process makes a choice of basic classifiers and methods for the structure and parameter learning for the collective classification.

2.2 Basic classifiers

The basic component in the document classification is a supervised probabilistic classifier of document elements. The classifier is trained with the local features and, for each unlabeled element x, it estimates conditional probabilities $P(y|x)$ for all possible labels y. In the following, we will primarily assume the maximum entropy classifier [Berger et al.(1996)]; other supervised classifiers, like the logistic regression or the multi-class SVM, can be used in the similar manner. With the constraints based on selected features $f_j(x, y)$, the maximum entropy classifier attempts to maximize the conditional likelihood $P(y|x)$ which is represented as a log-linear function $P(y|x) = \frac{1}{Z(x)} exp \sum_j \lambda_j \cdot f_j(x, y)$, where the parameters λ_i are learned from the training corpus and $Z(x)$ is a normalizing factor which ensures that all the probabilities sum to 1.

In layout-oriented documents, the classifier disposes a set of local features composed of different types of information. *Content features* express properties on text in the element x, e.g. $f_1(x, y)$ = 1 if y=**title** and text in x has only uppercase characters, 0 otherwise. *Structural features* capture the tree context surrounding an element, e.g. $f_2(x, y)$= 1 if y=**section** and x's father node has a left brother, 0 otherwise. *Attributes features* capture the values of node's attributes in the source tree, e.g. $f_3(x, y)$= 1 if y=**title** and the value of the *font* attribute of x is *times*, 0 otherwise.

2.3 Link features

A link feature $f_{\mathcal{P}}(x, y, y')$ captures the relationship between an element x, its label y and the label y' of a $\mathcal{P}$-node in the document structure, where $\mathcal{P}$ is a path in the structure from y to y'. For example, for a line x we may define $f_{\mathcal{P}}(x, y, y')$= 1 if $g(x)$ and y=**section** and y' is *metadata*, 0 otherwise, for some characteristics $g(x)$ of the element x and path $\mathcal{P}$ from a `LINE` to the corresponding `PAGE` node ($\mathcal{P}$="`ancestor::PAGE`" in the XPath language). The entire set of link features are generated by the template C which is a triple $(filter, path, g)$ where $filter$ is (x.type=`LINE`) and $path = \mathcal{P}$. In the similar manner, link features may be defined over links between elements linked with path $\mathcal{P}$.

The link features capture relationships between element labels and often form arbitrary graph structures. The Markov Random Fields (MRF) [Lafferty et al.(2001)] can address arbitrary undirected graphs, however finding exact solutions is computationally intractable in MRF. So we use an approximation technique of dependency networks and their relational extensions for typed data [Neville and Jensen(2007)]. Furthermore, when the conventional Gibbs sampling converges slowly, we consider an alternative method. We follow [Kou and Cohen(2007)] in considering the stacking principle for integrating link features $f(x, y, y')$ in the basic classifier. When a true value y' is unavailable, it is replaced by estimation $\hat{y'}$. In turn, the estimation model for $\hat{y'}$ is trained on pairs (x, y') in the available training corpus.

3 Dependency networks

In the input XML tree, local features are defined between input node charactertics $g(x)$ and node labels y. The link features are defined between each pair of neighbor `LINE`s and `PAGE`s, as well as between every `LINE` and the corresponding `PAGE` node.

The 2-layered representation G induces the *relational dependency network* used in the collective classification. Informally, the network includes the upper layer in G extended with extra-layered links. Formally, nodes in the network are input nodes x and all output nodes in Out-layer to be annotated with labels y (`LINE`s and `PAGE`s in Figure 2). Two types of undirected arcs are induced by static and link features defined over the input elements. An arc in the network links a node x to node y if one or multiple local features are defined over the pair (x, y). Similarly, an arc in the network links a node y to node y' if a link feature is defined over the pair (y, y'). For any node y in the dependency network, we denote $\mathbf{y}_{nb}$ the vector or neighbor nodes of y.

3.1 Gibbs sampling

The Gibbs sampling method [Bishop(2006)] is a Markov Chain Monte Carlo algorithm used when the inference is intractable because of the graph complexity. The Gibbs sampling proceeds by replacing the value of one of the variables y by a value drawn from the distribution of y conditioned on the values of the remaining variables. In other words, it replaces y by a value drawn from the distribution $p(y|\mathbf{y}_{nb}, \mathbf{x})$. The sampling procedure is repeated by cycling through the variables in $\mathbf{y}$ and choosing the variable $yinbfy$ to be updated at each step. The initial variable values $\mathbf{y}^0$ are provided by the classifiers trained with the local features only. At iteration $j, j = 1, \dots, J$, we sample y_m as follows:

$$y_m^{j+1} \sim p(y_m | y_1^{j+1}, \dots, y_{m-1}^{j+1}, y_{m+1}^{j}, \dots, y_M^{j}, \mathbf{x}), \tag{1}$$

where M is the total number of variables.

In the Gibbs sampling, the update of one variable on its neighborhood is computationally simple. However the convergence may be very slow, due to the dependencies between consequent samples; this forms the so-called *slow mixing* effect. To partially remedy the slow convergence of Gibbs sampling, we consider also its blocking variant [Jensen et al.(1995)]. The *blocking Gibbs sampling* tries to make the variable mixing faster and increases the number of variables sampled at each step. It visits variables simultaneously by groups, where one group of variables corresponds to a component in the network. The union of components comprise all variables but one one variable may be in several components. When a component is visited, it is sampled conditionally on the neighborhood of all its variables. One iteration of blocking Gibbs sampling is completed when all components are visited by the algorithm. Like in (1) for variables y, in the blocking Gibbs sampling, we sample a component c_m at iteration $j, j = 1, \dots, J$, as follows:

$$c_m^{j+1} \sim p(c_m | c_1^{j+1}, \dots, c_{m-1}^{j+1}, c_{m+1}^{j}, \dots, c_C^{j}, \mathbf{x}), \tag{2}$$

where C is the total number of components.

3.2 Stacked dependency networks

Stacked generalization is a general method of using a high-level model to combine lower-level models to achieve a higher predictive accuracy. The high-level model is built using the extended training set (x, $\hat{y}$, y), where $\hat{y}$ is a one or multiple low-level model predictions for labeling x with y. Recently, Cohen et al. extended the stacking to the Web page classification [Kou and Cohen(2007)]. We apply the stacked generalization method to the relational dependency networks and consider it as an alternative to the Gibbs sampling.

The collective annotation of two elements with labels y and y', in the presence of $\mathbf{x}$, is given by the joint conditional probability $p(y, y'|\mathbf{x})$. When y and y' are not independent, the exact solution assumes the enumeration of all possible annotation of y and y':

$$p(y, y'|\mathbf{x}) = \frac{p(y, y', \mathbf{x})}{\sum_{(y,y')} p(y, y', \mathbf{x})} = \frac{1}{Z} exp \sum_j \lambda_j f_p^j(y, y', \mathbf{x}). \tag{3}$$

The RDN approximates the exact solution by factorizing the joint probability and injecting the estimations as follows:

$$p(y, y'|x) \approx p(y|\hat{y}', \mathbf{x}) p(y'|\hat{y}, \mathbf{x}), \tag{4}$$

where $\hat{y}'$ and $\hat{y}$ are estimations of y' and y, respectively.

In the stacked version of RDNs, we are stacking a number of approximation models; in each model the collective annotation $p(y, y'|\mathbf{x})$ in the dependency network is approximated by $2 \cdot l$ models $p(y^i|\hat{y}'^{i-1}, \mathbf{x})$ and $p(y'^i|\hat{y}^{i-1}, \mathbf{x})$, $i = 1, \ldots, l$, where $\hat{y}^{i-1}$, $\hat{y}'^{i-1}$ are predictions for y and y' made on the previous level model, l is the stack size.

In the general case of the dependency network G, we denote $\mathbf{y}_{nb}$ all the neighbors y_{nb} the given y is linked with link features to. Then, the stacked RDN approximation factorizes the joint distribution over $\mathbf{y}$ on the each level of the stack as follows:

$$p(\hat{\mathbf{y}}^i|\mathbf{x}) = \prod_{y^i \in \mathbf{y}^i} p(y^i|\hat{\mathbf{y}}_{nb}^{i-1}, \mathbf{x}). \tag{5}$$

The stacking method constructs samples of ($\mathbf{x}$,$\hat{\mathbf{y}}$) where $\hat{\mathbf{y}}$ is a vector of class predictions for $\mathbf{x}$. To avoid the mismatch between data used in the training set and the testing set, the cross-validation principle should be applied *inside* the learning algorithm. The next section presents details of the learning and inference algorithms.

3.3 Supervised learning on stacked RDNs

If the input documents have nodes of different types, so does the dependency network, one type per relation. In the document structuring example, two types of variables (output nodes) refer to two different types, `LINE`s and `PAGE`s. Each node type $t \in T$ is associated with the proper class set Y_T and is characterized with an associated feature set. Consequently, learning a model for the input documents requires training T type-associated models, one model per type. Below, the learning and inference algorithms describe stacking algorithms for multi-type data.

Learning algorithm. Given a training set $S_t^0 = (\mathbf{x}_t; \mathbf{y}_t) = \{(x_t, y_t)\}$ for each type $t \in T$, a basic learning method M, a cross-validation parameter K, a stack size l.

Split S_t^0 into K equal-sized disjoint subsets $S_{t,1}^0, \ldots, S_{t,K}^0$.

for $i = 1, \ldots, l$ **do**
  for $k = 1, \ldots, K$ **do**
    for each type $t \in T$ **do**
      1. Get function F_k^i by training the basic method M on the set $S_t^{i-1} - S_{t,k}^{i-1}$
      2. Construct an extended dataset $S_{t,k}^i$ of instances $\{(x_t, \hat{\mathbf{y}}_{nb}^i), y_t\}$ by extending each x_t, $(x_t, y_t) \in S_{t,k}^0$ with estimations of neighbor nodes using function F_k^i, $\hat{\mathbf{y}}_{nb}^i = F_{t,k}^i(x_t, \mathbf{y}_{nb}^{i-1})$
  Compose $S_t^i = \cup_{k=1}^K S_{t,k}^i$
return functions F_t^i by training the basic method M on the sets S_t^i,
  $i = 1, \ldots, l, t \in T$.

The algorithm generates $|T| \cdot K \cdot l$ functions $F_{t,k}^i, k = 1, \ldots, K, i = 1, \ldots, l, t \in T$ to produce the extended datasets for all levels and types and returns $l \cdot |T|$ functions F_t^i used in the inference.

Inference algorithm. Given an input document containing elements of different types, $\mathbf{x}^0 = \{\mathbf{x}_t^0\}, t \in T$.
for level $i = 1, \ldots, l$ **do**
  for each type $t \in T$ **do**
    1. Produce estimation for i-level $\hat{\mathbf{y}}_t^i = F_t^i(\mathbf{x}_t^{i-1})$
    2. Produce an extended set $\mathbf{x}_t^i = \{(x_t^0, \hat{\mathbf{y}}_{ng}^i)\}$
return the estimation for the last level $\hat{\mathbf{y}}_t^l = F_t^l(\mathbf{x}_t^l)$ for all $t \in T$.

The inference algorithm is linear in the size of the number of output variables/nodes The complexity of inference algorithm is $O(l \cdot |T| \cdot |\mathbf{y}|)$, where $|\mathbf{y}|$ is the number of output variables/nodes.

4 Evaluation

We now evaluate the stacked RDNs and compare it to Gibbs sampling methods. The lower bound in our evaluation is the *Baseline* method, where the link features are ignored and the prediction models are trained using the local features only. As the upper bound for the stacking approximation, we may consider the *Exact* solution according to (3). However, when the inference of exact solutions is intractable because of the large size of input documents and complexity of dependency networks, we consider another upper bound methodology. The *Upper* bound solution is one where an oracle is assumed to correctly predict the neighbor classes. In our implementation, the oracle-based models are built with the extended training sets where the true neighborhood values $\mathbf{y}_{nb}$ augment explicitly the extended datasets for each y.

We then compare *Stacking* to several variants of the Gibbs sampling method. The performance of Gibbs sampling depends on the order of visiting variables $y \in \mathbf{y}$. We test two visiting policies. In the *Gibbs-random*, variables y are visited randomly. In the *Gibbs-order*, the variables are visited according to the reading order of elements in the document. Finally, we evaluate the ordered Blocking-Gibbs variant. We run different tests to compare *Baseline*, *Upper*, *Stacking*, Gibbs and *Exact* methods. The basic method M used in *Stacking* is the maximum entropy classifier. Below we report some results of tests applied on two different collections.

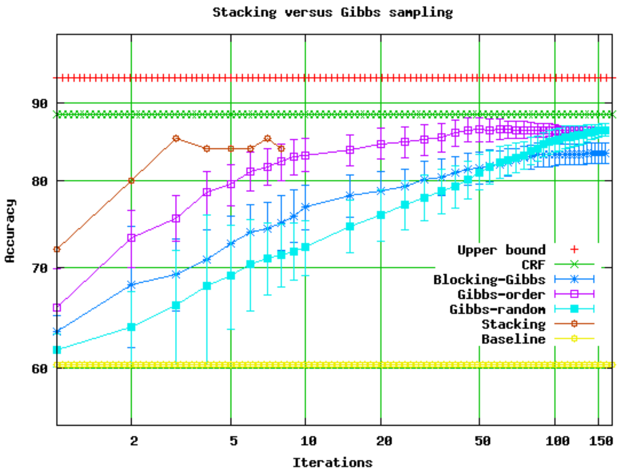

Figure 3: Inference with *Stacking* and Gibbs sampling methods.

Abstract annotation. The first collection is a set of 28 abstract sequences from the collection whose elements are annotated with 6 metadata classes. Each abstract is represented as a sequence of lines. In the dependency network generated for the sequences, the element's neighborhood is given by its left and right elements. Due to the chain dependency network, we can deploy the CRF++ package [1] to obtain the exact solutions.[2] For all tested methods, we estimate the accuracy of element annotation.

In the stacking methods, the stack size is varied from 1 to 8 and 5-folding cross-validation is used; the inner cross-validation parameter K is set to 2. In the first set of tests, we run *Stacking* method on the sequence collection with the basic chain structure ($\mathbf{y}_{nb}$) includes y's left and right brothers) and compare it to *Upper*, *Independent* and *Exact* solutions.

The model training is fast with both *Stacking* and Gibbs methods. Instead, for the

[1] CRF++: Yet another CRF toolkit, `http://crfpp.sourceforge.net/`.
[2] The inference with the CRF is $O(N \cdot Y^2)$ where Y is the number of different classes in $\mathbf{y}$.

inference, as Figure 3 shows, *Stacking* reaches the maximum accuracy in 3-4 iterations, while it takes 3 to 20 times more iterations to Gibbs sampling. Among the Gibbs sampling methods, the *Blocking-Gibbs* performs better than other variants.

	Precision	Recall	F1
Upper	*94.87*	*91.00*	*92.89*
Stacking l=5	83.99	75.38	**79.46**
Stacking l=4	83.78	74.60	78.93
Stacking l=3	83.99	**75.38**	79.46
Stacking l=2	83.98	74.87	79.17
Stacking l=1	**84.10**	74.73	79.14
Gibbs	83.94	75.15	79.23
Baseline	*77.38*	*69.87*	*73.44*

Table 1: Evaluation for the section segmentation case.

Section segmentation. The second collection is composed of 13 large technical documents, accounting in total for 7,073 pages and 275,299 lines. The document structuring task has been presented in Sections 2; it targets annotating LINEs and PAGEs in the layout-oriented documents which correspond to the beginning of sections and the section titles; there are 1058 such lines and 501 such pages in the collection. The link features of the target dependency network have been presented in Section 2.3. In all tests, we report the precision, recall and F_1 measures, separately for each type and jointly for the entire set. As in Figure 2, $\mathbf{y}_{nb}$(LINE) includes two neighbor LINEs and a PAGE the given line is a part of; $\mathbf{y}_{nb}$(PAGE) includes two neighbor PAGEs and all LINEs on the page.

	LINEs	PAGEs	LINEs+PAGEs
Upper	*87.43*	*100.00*	*92.86*
Stacking l=1	78.74	79.32	78.98
Gibbs	77.95	80.93	79.23
Baseline	*75.04*	*70.79*	*73.01*

Table 2: Evaluation for different element types.

When we train stacking models on the section collection, the stack size l vary from 1 to 5 and 5-folding is used for both internal and external cross-validations. Table 1 reports results for the *Baseline*, *Upper*, *Gibbs* and *Stacking* methods (no exact solution is possible because of the graph size and complexity). As the table shows, we observe no significant difference between models with different stack sizes. Unlike the first collection, the Gibbs sampling converges fast with Finally, Table 2 reports $F1$ values for different element types with the stacking inference and compares them to the *Baseline* and *Upper* bounds. The clasification of PAGEs takes the maximum benefit from the known neigbourhood in the *Upper* bound; with the *Stacking*, $F1$ values are comparable for both types.

Analysis. The analysis of series of tests on both collections allowed us to make two major conclusions. First, the stacking inference is comparable to Gibbs sampling in the RDNs but additionally shows an important reduction in the inference time, despite the extra work needed to train the stacking models. Second, in all the tests, the structure of the dependency networks have been set manually, through the user-based definitions of the link features. We are now looking for the RDN structure inference, by the automatic discovery of important long-distance correlations between elements and their coupling with the stacking generalization.

For a given document annotation task, the optimal solution should take into account the variability of documents and the complexity of structuring task. If a sequential ot tree-like representation of documents is possible, the exact inference with CRF should be the first choice; such a conclusion being consistent with other deployments of CRF for the document analysis [Shetty et al.(2007)] and handwriting recognition [Feng and McCallum(2007)]. Otherwise, the RDNs are a viable alternative for complex structuring problems. The convergence of Gibbs sampling may vary considerably from one case to another, and in most cases, the stacking generalization may help reduce the inference time.

5 Conclusion

We have addressed the problems of annotating and segmenting the larges layout-oriented documents. We approach the document structuring by the mapping from input data to variables on typed nodes and links which are subjects of the collective classification. We apply the principle of relational dependency networks defined by local and link features that capture relationships between input elements and their labels as well as between elements' labels. We then extend RDNs with the stacking generalization principle which ensures the fast inference as compared to the Gibbs sampling.

References

[Berger et al.(1996)] Berger, A. L., Pietra, S. D., Pietra, V. J. D.: "A maximum entropy approach to natural language processing"; Computational Linguistics; 22 (1996), 1, 39–71.

[Bishop(2006)] Bishop, C. M.: Pattern Recognition and Machine Learning; Springer, 2006.

[Chanod et al.(2005)] Chanod, J.-P., Chidlovskii, B., Dejean, H., et al.: "From legacy documents to xml: A conversion framework"; Proc. European Conf. Digital Libraries; 92–103; 2005.

[Dejean and Meunier(2005)] Dejean, H., Meunier, J.-L.: "Structuring documents according to their table of contents"; DocEng'05: Proc. ACM Symp. on Document Engineering; 2005.

[Feng and McCallum(2007)] Feng, M. R., S., McCallum, A.: "Exploring the use of conditional random field models and hmms for historical handwritten document recognition"; Proc. 2nd Intern. Conf. on Document Image Analysis for Libraries, DIAL'06; 30–37; 2007.

[Jensen et al.(1995)] Jensen, C., Kong, A., Kjaerulff, U.: "Blocking Gibbs sampling in very large probabilistic expert systems"; Intern. Journal of Human Computer Studies; 42 (1995), 647–666.

[Kopec and Chou(1994)] Kopec, G. E., Chou, P. A.: "Document image decoding using Markov source models"; IEEE Trans. Pattern Anal. Mach. Intell.; 16 (1994), 6, 602–617.

[Kou and Cohen(2007)] Kou, Z., Cohen, W. W.: "Stacked graphical models for efficient inference in Markov random fields"; Proc. SIAM Data Mining; 2007.

[Lafferty et al.(2001)] Lafferty, J., McCallum, A., Pereira, F.: "Conditional random fields: Probabilistic models for segmenting and labeling sequence data"; ICML '01: Proc. 18th Intern. Conference on Machine learning; ACM Press, New York, NY, USA, 2001.

[Liang et al.(2005)] Liang, P., Narasimhan, M., Shilman, M., Viola, P.: "Efficient geometric algorithms for parsing in two dimensions"; International Conference on Document Analysis and Recognition (ICDAR); 2005.

[Mao et al.(2003)] Mao, S., Rosenfeld, A., Kanungo, T.: "Document structure analysis algorithms: a literature survey"; Proc. SPIE Electronic Imaging; volume 5010; 197–207; 2003.

[Neville and Jensen(2007)] Neville, J., Jensen, D.: "Relational dependency networks"; Journal of Machine Learning Research; 8 (2007), 653–692.

[Shetty et al.(2007)] Shetty, S., Srinivasan, H., Beal, M., Srihari, S.: "Segmentation and labeling of documents using Conditional Random Fields"; Proceedings of SPIE; 2007.