Low level information extraction

a Bayesian network based approach

Remco R. Bouckaert rrb@xm.co.nz, remco@cs.waikato.ac.nz
Xtal Mountain Information Technology & Computer Science Department, University of Waikato, New Zealand

Abstract

In this article, a contribution is made to information extraction and Bayesian network learning motivated by two practical information extraction tasks.

It is shown that some information extraction tasks can be approached as a classification problem where the text is split in tokens and each token is assigned a class. Hidden Markov models are a popular formalism for this task, however they do not deal with tokens having a set of attributes instead of a single one. A new algorithm for this task is presented using various Bayesian networks architectures that deals with multiple attributes per token. Experiments suggest most Bayesian networks architectures perform better than naive Bayes in our problem domain.

Hopefully, this article helps in making Bayesian networks more accessible for the information extraction community.

1. Introduction

Information extraction is the task of finding specific pieces of information from unstructured documents. Though it is generally known that the internet contains large amounts of unstructured documents on which information extraction tasks can be performed, it seems to be less well publicized that large organizations like publishing companies, government departments, airplane manufacturers, etc. have an enormous amount of unstructured documents as well.

With the exploding amount of information captured in an ever increasing number of documents, the task of finding the information required becomes harder and harder. The popularity of extensible markup language (XML)[1,2] in the (publishing) industry[3] indicates that many aspects of a document can be captured in a structured format, which assists in making documents more accessible to queries and hence information gathering tasks. When there is a large collection of documents, they tend to be written by a large variety of people, each with their own styles. This makes the task of creating a structured document from an unstructured one a big challenge. Structuring documents can be split in two subtasks:

- Determine the high level structure, that is, determine where the sections are with their headings, which part is the reference section, what is a table, etc.
- Determine the low level structure, that is, given that you know a piece of text contains an affiliation, determine the individual elements of the affiliation like, for example, institute, street address, post box, city, zipcode, state, and country.

In this context, for both high and low level structure retrieval, information extraction is the task of finding out which part of the text contains particular pieces of information.

Information extraction is already successfully applied in areas such as job advertisement collection on the internet[4], scientific article collection from the internet [16, 17] and datamining medline (an internet repository of medical related documents) for gene interactions[5]. DEADLINER [13], GeoSearch [9] and

[1] The World Wide Web Consortium. Extensible Markup Language (XML). http://www.w3c.org/XML/

[2] All URLs mentioned in this article were accessed and available at 21 April 2002.

[3] XML Industry Support. http://xml.coverpages.org/xmlSupport.html[2]

[4] Job search - find jobs, employment, & careers at FlipDog.com. http://www.flipdog.com[2]

[5] Tor-Kristian Jenssen. Pubgene: Datamining Medline for Gene Interactions. http://www.pubgene.org[2]

HPsearch[6] are other examples of internet based information extraction applications.

To perform information extraction tasks on completely unstructured documents without understanding the content completely would be impossible. Since there is quite some way to go in document understanding, a simpler task will be considered here. In this article, we will concentrate on low level structure discovery, assuming the high level extraction has been done already.

Commercial packages are available to support information extraction based on regular expression parsing (e.g. XMLCities suite, TextTerity's TextCafe and ItemField's ContentMaster). They come with fancy graphical user interfaces that support entering regular expressions in a less cumbersome format than Perl syntax. However, in practice, considerable effort is required to use regular expressions for this task, because domain knowledge needs to be encoded. This tends to be a cumbersome task since the domain knowledge is in general not readily available. Several approaches based on hidden Markov models [5, 10, 18] have been undertaken, but these approaches are limited by the effort required to learn the Markov model.

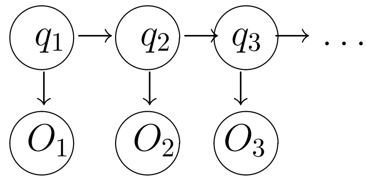
A new approach is proposed based on the observation that information extraction can in some cases be interpreted as a classification task when the dimensionality of the tokens can be reduced. Every token (word or sequence of words) is assigned a class based on the lower dimensional attributes calculated for the token. For example in the address recovery example just mentioned before every word can be classified as institute, street address, post box, city, zipcode, state, or country. Bayesian networks, and in particular networks restricted to some specific network architectures are well suited to this task [4, 11].

In this article, first we introduce Bayesian networks together with terminology and notations used in the remainder of this article. Then, we present an algorithm for performing the subtask of structure extraction from a document. Case studies are presented, which show the properties of the algorithm on some real world problems. We finish with concluding remarks and some directions for future research.

2. Bayesian networks

Informally, a Bayesian network consists of a graph and a set of local probability distributions: one for each of the nodes in the graph. The graph is directed

Figure 1. Bayesian Network representing a HMM



and acyclic, that is, there are no directed paths starting and ending at the same node. The direction of the arrows in the graph are sometimes interpreted as causal influences, hence the alternative name causal network (but there are a lot more alternative names). A Bayesian network represents a probability distribution over the nodes in the graph, which can be used to perform interesting calculations, in particular classification.

2.1. Terms and definitions

We define the terms and phrases more formally now. Let U be a set of variables $u_1, \ldots, u_k$, $k \geq 1$. A *classification variable*, normally identified by c, is a variable in U for which we are interested in predicting the outcome given the values of the other variables $U \backslash \{c\}$. The other variables are *attribute variables*.

A graph G is a pair (U, E) where U is a set of *nodes* and E is a set of pairs (u, v) called *edges*, $u, v \in U$, $u \neq v$ Edges can be directed edges, called *arrows* or undirected edges called *lines*. A *path* in a graph is a sequence of nodes $v_1, \ldots, v_k$, $k > 1$, such that $\forall_{1 \leq i < k}(v_i, v_{i+1}) \in E$. A *directed cycle* is a path where $v_1 = v_k$. A *directed acyclic graph* (DAG) G over a set of nodes U is a directed graph without directed cycles.

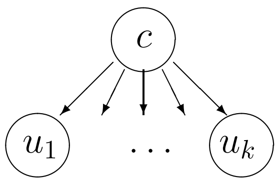
A *Bayesian Network* B over a set of variables U is a pair (B_S, B_P) where B_S is a DAG (U, E) $n = |U|$, and B_P is a set of conditional probability tables $P(v_i|\pi_i)$, $1 < i \leq n$. The set of probability tables spans up a probability distributions over U:

$$P(U) = \prod_{i=1}^{n} P(v_i|\pi_i)$$

This distribution can be used to calculate interesting queries. In particular for a set of variables $A \subset U$ and the observed values of another set of variables $B \subset U$, the probability $P(A|B)$ and the most likely configuration $argmax_A P(A|B)$ of A can be calculated using

[6]HomePageSearch http://hpsearch.uni-trier.de/hp/[2]

Figure 2. Naive Bayes as Bayesian Network



efficient algorithms [15, 21].

A Hidden Markov Model (HMM) λ over a state space $V = \{S_1, \dots, S_k\}$ and observation space $O = \{O_1, \dots, O_m\}$ is a triple $\lambda = \{\pi, a, b\}$. Here $\pi = P(q_1 = S_i)$ is the initial probability over states at $t = 1$. Further, $a = P(q_t = S_j | q_{t-1} = S_i)$ the transition probability for state S_j given the previous state. And finally, $b = P(O_t = O_j | q_t = S_i)$ the probability that observation O_j is generated if the state q_t is S_i. Figure 1 illustrates that a HMM can be interpreted as a special case of a Bayesian network with a node for each token and a node for each state. The probability tables are defined as $P(q_1) = \pi$, $P(q_{i+1}|q_i) = b$ and $P(O_i|q_i) = a$ for $i \geq 1$.

2.2. Naive Bayes classifier

A Naive Bayes classifier is a Bayes network with the following structure: the classification variable c has outgoing arrows $c \rightarrow u$ to each of the attribute variables $u \in U \backslash \{c\}$ and no other arrows are added. This network structure will be called *naive Bayesian network*. Figure 2 shows the network structure of a Bayesian network that represents a naive Bayes classifier. Because the structure is static, there is no need to perform structural learning, only probability tables $P(c)$ and $P(u|c)$ need to be assessed.

2.3. TAN classifier

A Tree Augmented Naive Bayes (TAN) classifier [4, 11] is a Naive Bayesian network with some extra arrows. The structure when the classifier node is removed is a tree structure (i.e. none of the nodes has two incoming arrows from non-classifier nodes). The structure can be learned by starting with a Naive Bayes network and hill climbing on a quality measure from there onwards (details follow in Section 3.1). The number of incoming arrows for a node is restricted to at most 2; 1 from the classifier node, and at most 1 from another node.

2.4. BAN classifier

A Bayes network Augmented Naive Bayes (BAN) classifier [4, 11] is also a Naive Bayesian network with some extra arrows, but the structure when the classifier node is removed is a full Bayesian network. The structure can be learned by starting with a Naive Bayes network and hill climbing from there onwards, without restricting the number of incoming arrows.

2.5. GBN classifiers

A General Bayesian Network (GBN) classifier is a classifier that does not have any restrictions on the structure of the network. Starting with an empty structure and hill climbing can be used to learn such a structure.

3. Information extraction

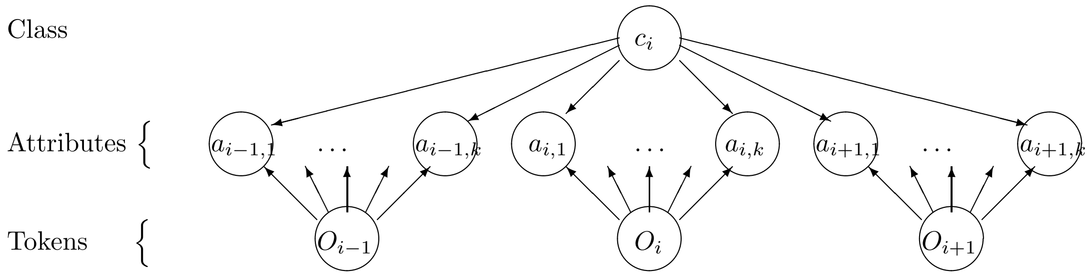
To retrieve low level structure, first of all a sentence need to be broken up in tokens. Then, each of the tokens need to be assigned a class. HMMs have been applied with success to the latter task [5, 10, 18]. The power of HMMs lies in taking into account the whole of the set of tokens to classify. The problem with hidden Markov models is learning the transition probabilities. Also, typically only a single property of a token is taken into account, though HMM generalizations like conditional random fields [14] and maximum entropy Markov models [19] overcome this problem.

However, a token can be assigned many properties that are not easily captured in a single variable. Examples are whether a token is all uppercase, whether it contains digits and the number of characters in the token. So, it is natural to calculate a set of properties for each of the tokens. HMMs do not deal with this situation. Bayesian networks are very suited to the task, but do not have the desirable property of taking into account the neighborhood. However, to do this it may be sufficient to just have an attribute for the position (e.g. with values first, middle, and last) and have the attributes of previous and next token available. Figure 3 shows the Bayesian network illustrating this. The only thing left is to learn is the Bayesian network structure among attribute nodes.

3.1. Bayesian network learning

In general, learning a Bayesian network $B = (B_S, B_P)$ consists of two steps: learning the structure B_S and learning the probability tables B_P. In this article, to learn B_S a search in the space of network structures is performed and a network is assigned a measure of quality based on the data. Various measures exist based on

Figure 3. Calculating attributes from tokens



Bayesian [12], minimum description length and other information criterion approaches [2]. These measures all behave more or less the same for larger datasets [2], so in the remainder the Bayesian approach is taken. This approach maximizes the probability of the network structure B_S given the data D and can be decomposed by the prior on the network structure $P(B_S)$ and the product of the quality measures $P(u_i, \pi_i|D)$ of the nodes and their parents:

$$P(B_S|D) \propto P(B_S) \prod_{i=1}^{n} P(u_i, \pi_i|D). \tag{1}$$

It is customary not to prefer any network structure when no extra information about the structure of the problem domain is available, so we assume $P(B_S)$ is constant and can be discarded. Because of decomposition (1), finding a good network structure now reduces to finding the parent sets of each node where only the expression $P(u_i, \pi_i|D)$ needs to be optimized. There is one caveat: because the network structure is an acyclic graph, before adding an arrow, it has to be tested that no cycles appear. A simple way to prevent this is ordering the nodes and allow arrows only from lower ordered nodes to higher ordered ones. The structure learning algorithm K2 [6] makes this assumption and for each node does hill climbing on the parent set. Algorithm B [3] does hill climbing without assuming an ordering, so for every arrow to be added an acyclicity test needs to be performed making it less efficient.

In the information extraction tasks we encounter, no data is missing. Consequently, the conditional probability tables can be estimated directly from the data once the structure of the network is known. If data were missing, an EM algorithm [12] (generalization of the Baum-Welch algorithm for HMMs) can be applied.

3.2. Markov Blanket classifiers

The initial structure of the network (empty or Naive Bayes) has a major impact on the quality of the learned network [4, 11]. This suggests that putting a bit more effort in analyzing the initial structure may pay. The reason that a Naive Bayes network performs better than an empty initial structure is that it is guaranteed that every node has an effect on the classifier node, however small.

This observation suggests that an initial network where every node is in the *Markov blanket*, that is the set of parents, children and parents of those children, of the classifier node may be a good starting structure, because only nodes in the Markov blanket influence the classification. The only problem left is finding such a network, that is, designing an algorithm that efficiently finds such a network.

The number of such networks is #ways to select a set of parents times #ways to select children from remaining nodes times #ways to assign remaining nodes to parents of children of classifier node, which is,

$$\begin{aligned} & \sum_{i=0}^{n} (\frac{n}{i}) \times \sum_{j=1}^{i} (\frac{i}{j}) \times j^{n-i-j} \\ \geq \quad & \sum_{i=0}^{n} (\frac{n}{i}) \times \sum_{j=1}^{i} (\frac{i}{j}) = 3^n. \end{aligned}$$

So, there is an exponential number of models to consider. A practical alternative is to learn a network structure starting with an empty network, apply K2 or B and test whether a node is in the Markov blanket of the classifier node. If it is not, add an arrow between the node and the classifier node. To guarantee acyclicity with K2 always an arrow from the classifier node to the attribute node can be added, while for B it has to be tested whether the attribute node is an ancestor of the classifier node or not. If it is, an arrow is added pointing to the classifier node, otherwise it points to the attribute node. The resulting Bayesian network will be called a *Markov blanket classifier*.

3.3. Optimizations

To reduce learning times and prevent overfitting, a bound on the maximum number of parents can be provided for the algorithms. Note that as a result, a BAN classifier with a restriction of having at most 1 parent will reduce to a TAN.

No effort was done to provide more sophisticated learning algorithms than straightforward hill climbing, also to reduce learning time. Obviously, this way a network with a better quality (according to the quality measure used) could be missed.

4. Case study 1: affiliations

A sub-problem in converting scientific documents written by various authors to a uniform XML format is the extraction of affiliations of the authors. To start with, a simple set or heuristic rules were applied. Those rules are of the form 'If position of the token is last, it is a country'. This algorithm was used to gain some understanding in the problem domain and assisted in creating training data. Also, because conventional technology was used with which the business environment was comfortable, it was easy to integrate in existing systems and make it maintainable.

Affiliations have the property that items are separated by commas, which allows for a simple token definition: any text between commas (or start or end of the affiliation string) is a token. We start with a description of the attributes calculated for each token, then the experimental setup follows, and finally results are presented and discussed.

4.1. Description of data

For each token, the following attributes are calculated:

- position: First, Mid, Last. Indicates what the position of the token is within the affiliation.
- IsInstitute: 0, 1. Indicates whether the token has one of the keywords "University", "Hospital", "Limited", "Institute", "Foundation", "Pharmaceuticals", "Universidad", "Universitario".
- IsDepartment: 0, 1. Indicates whether the token has one of the keywords "Department", "Division", "Laboratory", "Branch", "Section", "School", "Unit", "Program", "Programme".
- IsStreetAddress: 0, 1. Indicates whether the token has one of the keywords "Street", "St.", "Strasse", "Court", "Lane", "Ln", "Road", "Rd", "Rue", "St.", "Avenue", "Ave", "av.", "Drive", "Dr", "Highway", "Hwy", "Way", "Parkway", "Boulevard", "Blvd", "Piazza", "Plaza", "Platz", "Room", "Floor", "Building", "Suite", "Via".
- IsPOBox: 0, 1. Indicates the token matches the regular expression /\bp\.?o\.?\s?bo?x?\b/ or /\bbox\b/ or /^\s*\bPB\b\s+[0-9\s]+$/.
- IsCity: 0, 1. Indicates the token is part of a list of cities.
- IsZipCode: 0, 1. Indicates the token matches /^[\-\sA-Z0-9]*$/
- IsState: 0, 1. Indicates the *following* token is part of a list of countries.
- IsCountry: 0, 1. Indicates the token is part of a list of countries.
- NrOfWords: 1, 2, 3, 4+. Counts the number of words in a token.
- HasNrs: 0, 1. Indicates the token contains at least one number.

The classifier is provided with the attributes for a token, plus the attributes for the previous token and next token. For the first (last) token, the previous (next) token attributes are all set to zero and position attribute is set to First (Last).

The classes are institute, department, city, street, pobox, country, zip, zipncity, state and netaddress. Note that 'zipncity' is a class where the token contains both a zipcode and a city, something that happens because regularly an affiliation is provided without comma between zipcode and city.

This makes a total of 34 attributes. The affiliations are sourced from articles of a pharmaceutical publisher. A total of 508 affiliations providing 2418 tokens are in the dataset. Affiliations are mixed in that some affiliations provide a summary of an affiliation, for example, "Childrens Hospital Medical Center, Division of Infectious Diseases, Cincinnati, Ohio, USA." and some provide a full postal address, for example "University of Melbourne, Parkville, Exercise Physiology & Metabolism Laboratory, Department of Physiology, VIC 3052, Australia".

The labeling was initially performed automatically using a simple set of rules. Then, the dataset was manually cleaned in which about 13% of the cases were relabeled.

Table 1. Average accuracy in percentage for affiliation extraction (std dev. in brackets).

max # parents	K2	K2 Naive	K2 Markov	B Naive	B Markov
0	30.4				
1 (Naive)	93.5 (1.56)	93.3 (1.60)	93.1 (1.61)	93.3 (1.60)	94.1 (1.45)
2 (TAN)	95.1 (1.48)	95.1 (1.39)	95.0 (1.46)	95.2 (1.20)	95.0 (1.22)
3 (BAN)	95.3 (1.31)	95.4 (1.28)	95.3 (1.30)	95.7 (1.25)	95.6 (1.32)
4 (BAN)	95.4 (1.25)	95.5 (1.30)	95.4 (1.26)	95.9 (1.18)	96.0 (1.30)
5 (BAN)	95.6 (1.23)	95.6 (1.25)	95.6 (1.22)	96.0 (1.28)	96.0 (1.29)

4.2. Experimental set-up

The algorithms were run ten times with ten fold cross validation, since ten fold cross validation tends to give a good indication how the algorithm behaves on new data. Since any misclassification leads to a manual clean up action, the classification accuracy of the model instead of the more customary precision and recall criteria is taken as quality measure. Therefore, in this particular application accuracy can be justified as the measure to maximize, though this is not always suitable in text classification problems [22, 24]. The algorithms are implemented in Java using Weka [23]. Algorithms are compared using a T test, which is known to have a somewhat elevated type 1 error [7] in this setting, but is conveniently available in Weka.

Table 1 shows results. The first column shows the maximum number of parents for a node with the network structure for Naive Bayes initialized networks between brackets. The following columns show the classification accuracy averaged over the ten runs for ten fold cross validation, so the numbers shown are calculated from a total of hundred runs for each algorithm. Columns labeled 'Naive' use a naive Bayes network as initial network structure, and columns labeled 'Markov' use Markov blanket optimization.

4.3. Discussion

When the bound on the number of parents a node can have is set to zero, the class assigned is always the majority class in the dataset, which is not a good predictor in our case, as the classification accuracy of 30.4% shows in Table 1. When the bound is set to 1 and the initial network is the naive Bayes network, no other arrows are added, and we get the result for the naive Bayes classifier. The classification accuracy tends to increase with the increase of the bound, but it levels off. In particular, naive Bayes does worse than any network structure that is more complex than the naive Bayes one (at 0.05 significance level with double tailed T test). In general algorithm B outperforms algorithm K2, which is to be expected due to the larger search space algorithm B visits. This comes at the cost of increased learning time.

Table 2 shows ranking of the algorithms by the number of times an algorithms is significantly better (according to a double tailed T test at 0.05 level) minus the number of times an algorithms is significantly worse. It confirms that Markov optimization is useful for learning classifiers since Markov optimized networks outperform networks that have no optimization at all (the GBNs at the bottom of the table).

The algorithm based on heuristic rules, which is currently part of a production system, had a classification rate of 2107/2418= 87.1%. As Table 1 shows, learning algorithms significantly outperformed the rule based one. One could claim that by designing better rules, a better result can be obtained. Though the automatically learned models have their attributes based on the experience gained with the rule based algorithm, the time and effort required to design such an algorithm is easily offset by the time and effort required to apply an automatically learned model.

5. Case study 2: citations

Another subproblem of document conversion is citation extraction. Here we consider sentences that contain complete citations. The goal is to retrieve citation information like for instance authors, title, date and journal from the string. In this case it is not as easy as for the affiliation problem to define tokens because there are several characters to separate items of interest. For example, authors and title can be separated by spaces or full stops, while volume and issue can be separated by braces. The approach taken here is to define tokens as words and use the token separator as an attribute.

Table 2. Ranking on affiliation data

Algorithm: Search algorithm	Max # parents	Optimization	total	wins	losses
B	5	Naive	22	22	0
B	5	Markov	21	21	0
B	4	Markov	21	21	0
B	4	Naive	20	21	1
B	3	Naive	12	16	4
B	3	Markov	11	15	4
K2	5	Markov	11	15	4
K2	5	Naive	11	15	4
K2	5	-	11	15	4
K2	4	Naive	6	11	5
K2	4	-	2	11	9
K2	4	Markov	1	10	9
K2	3	Naive	1	10	9
K2	3	-	0	9	9
K2	3	Markov	-3	8	11
B	2	Naive	-6	7	13
K2	2	Naive	-7	7	14
K2	2	-	-10	6	16
B	2	Markov	-11	5	16
K2	2	Markov	-13	5	18
B	1	Markov	-16	4	20
K2	1	-	-18	3	21
B	1	Naive	-21	1	22
K2	1	Naive	-21	1	22
K2	1	Markov	-24	0	24

5.1. Description of data

For each of the tokens, the following attributes were generated:

- position: First, Mid, Last. Indicates what the position of the token is within the citation.
- IsAllUpper: 0, 1. Indicates all characters are upper case alphabetical.
- IsAllLower: 0, 1. Indicates all characters are lower case alphabetical.
- IsAllAlpha: 0, 1. Indicates all characters are alphabetical.
- IsCapitalized: 0, 1. Indicates the word starts with an uppercase alphabetical character and continues with the remainder in lower case alphabetical characters.
- IsAllNum: 0, 1. Indicates all characters are digits.
- HasNum: 0, 1. Indicates there is at least on digit.
- LengthOver2: 0, 1. Indicates whether the length is more or less than 2 characters.
- Separator: dot, comma, colon, semi-colon, question mark, space. Separator is the character after the current token.

Note that unlike for the affiliation case, no domain specific attributes like IsInstitute and IsDepartment were generated. The target class values are journal, authors, number, date, inbook, organisation, volume, title, note, pages, editor, edition, institution, chapter, publisher, address, and wrapper. The value wrapper is used for anything not in any of the other categories, for example text like 'In' to indicate a publication is part of a book like citation [10] in this article. Another approach is to extract authors, and year information from a citation and use that for matching against a database. This was motivation for creating another dataset with a different class attribute having values of authors, title, date and wrapper only. Any of the values that are not authors, title or date are mapped to the value 'wrapper'.

The data used for training and testing is the TEX bibliography archive for typesetting[7]. This archive has quite a few omissions in for example page numbers, however punctuation is reasonably accurate through BIBTEX processing, which is a realistic reflection of author provided manuscripts. The bibliography items are typeset using a plain BIBTEX style that is updated to insert tags with class information. Ascii text is extracted from the thus generated postscript file (some normalization of characters like ligatures is applied) and attributes calculated with a Perl script. The resulting dataset has 16316 cases with 26 attributes.

5.2. Experimental set-up

Again, ten runs of ten folds cross validation was used to generate test results. Table 3 shows the results where the first row contains the dataset: 'full' for the full set of values for the class, 'truncated' for the four valued class, and 'with class' is added if the class of the previous token is provided in the dataset (extending the dataset to 27 attributes). The first column shows the algorithm as in Table 1 and the other columns the averaged results for ten folds where the structure is learned using K2 or B.

[7]Collections of BIBTEX references can be downloaded from http://liinwww.ira.uka.de/bibliography/index.html[2].

Table 3. Average accuracy in percentage for citation extraction (std dev. in brackets).

dataset	Full		Full with class		Truncated		Truncated with class	
max # parents	K2	B	K2	B	K2	B	K2	B
0	38.2				41.1			
1 (Naive)	60.8(1.06)	60.8(1.06)	85.6(0.80)	85.6(0.80)	69.5(1.06)	69.5(1.06)	84.5(0.97)	84.5(0.97)
2 (TAN)	66.6(1.01)	66.6(1.05)	89.0(0.75)	89.2(0.75)	74.2(1.02)	73.9(1.15)	88.6(0.91)	90.6(0.73)
3 (BAN-3)	68.6(0.98)	69.4(0.94)	89.7(0.76)	90.4(0.76)	77.7(1.09)	77.9(1.58)	91.8(0.80)	91.9(0.69)
4 (BAN-4)	70.8(0.86)	71.1(0.96)	91.0(0.71)	92.4(0.59)	78.8(1.02)	78.6(1.07)	93.0(0.63)	94.1(0.60)
5 (BAN-5)	71.4(0.91)	71.7(0.92)	91.0(0.76)	92.6(0.59)	79.2(1.01)	78.7(1.08)	94.1(0.54)	94.4(0.60)

5.3. Discussion

Table 3 shows (max # parents = 1) the most prominent class makes up 38.2% of the full class and 41.1% of the truncated class. Again, any network structure more sophisticated (max # parents > 1) than naive Bayes (max # parents = 1) does better. Also, algorithm B outperforms K2 in most cases due to its larger search space.

The resulting classification accuracies as shown in Table 3 are encouraging considering that the attributes calculated for the tokens are rather standard and contain no domain specific information unlike the attributes for the affiliation classification problem.

The classification accuracy increases to acceptable levels when the class of the previous token is added. This suggests that citation identification may be assisted with classifying all tokens in a single task (like HMMs do) instead of independent from each other. Dynamic Bayesian networks [8] are similar to HMMs in that they define a network structure for each token with dependencies on the previous tokens attributes, and thus span up a distribution over all attribute and class variables for a sequence of tokens. However, they have the benefit over HMMs in that they allow for more flexible network structures allowing among others handling of multiple attributes per token. Efficient algorithms exist for calculating the configuration of class values with highest probability given the attributes for each of the tokens $argmax_{c_1 \ldots c_n} P(c_1 \ldots c_n | x_{1,1} \ldots x_{n,k})$.

6. Conclusions

For two practical problems, affiliation identification and citation parsing, it was demonstrated that Bayesian networks can help perform information extraction tasks. A handcrafted algorithm (part of a production system) developed with traditional software engineering techniques was outperformed both in terms of classification accuracy and development time. The problems allowed for handling the information extraction task as a classification task by tokenizing the input and generate various attributes for each of the tokens, which are used as input for the classification algorithm. Even defining standard attributes (e.g. "is all numeric", "is all upper case") as for the citation problem produce reasonable results, while putting more effort in the definition of attribute gives very useful classification accuracy.

A new Bayesian networks architecture, the Markov Blanket classifier, was introduced. This architecture allows for learning a network structure using standard algorithms and post-processing the learned network to ensure every variable has an impact on the classification. Experiments suggest the performance is better than plain Bayesian networks and comparable to networks initialized as naive Bayes network.

The observation that Bayesian networks can help in information extraction tasks hopefully inspires further research in this area, which may exploit an already available rich theory on graphical models. In the case study on citation classification it was shown that the classification accuracy for a token increases considerably when the class of the previous token is added. Dynamic Bayesian networks [8] are similar to HMMs in that they define a network structure for each token with dependencies on the previous tokens attributes, and thus span up a distribution over all attribute and class variables for a sequence of tokens. Further research in applying dynamic Bayesian networks for information extraction looks therefore highly promising.

Acknowledgements

I thank the anonymous referees for their useful hints and pointers to some related work.

References

[1] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html[2]

[2] R.R. Bouckaert. Properties of Bayesian Belief Network Learning Algorithms. Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference. 102–109. 1994

[3] W.L. Buntine. Operations for Learning with Graphical Models. Journal of Artificial Intelligence Research 2: 159–225, 1994

[4] J. Cheng and R. Greiner. Comparing bayesian network classifiers. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99), 101–107. Morgan Kaufmann Publishers, August 1999.

[5] J. Connan and C.W. Omlin. Bibliography Extraction with Hidden Markov Models, Technical Report US-CS-TR-00-6, 24 February 2000, Department of Computer Science, University of Stellenbosch

[6] G.F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning 9: 309–347, 1992

[7] T. G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation, 10(7) 1895–1924, 1998

[8] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. AAAI. 524–528, 1988

[9] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of web resources. In 26th International Conference on Very Large Databases, VLDB 2000, Cairo, Egypt, September 10-14 2000. http://www.northernlight.com/geosearch.html[2]

[10] D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In Proceedings of Workshop on Machine Learning and Information Extraction (AAAI-99), Orlando, FL, July 1999

[11] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian Network Classifiers. In Machine Learning 29:131–163, 1997

[12] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. Machine Leariníng, 20(3):197–243, 1995

[13] A. Kruger, C.L. Giles, F. Coetzee, E. Glover, G. Flake, S. Lawrence, and C. Omlin. DEADLINER: Building a new niche search engine. In Ninth International Conference on Information and Knowledge Management, CIKM 2000, Washington, DC, November 6-11 2000

[14] J. Lafferty, A. McCallum and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. ICML-2001

[15] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems (with discussion). Journal of the Royal Statistical Society B, 50:157–224, 1988

[16] S. Lawrence, K. Bollacker, L. Giles, NEC Research Institute. ResearchIndex: The NECI Scientific Literature Digital Library. http://citeseer.nj.nec.com/[2]

[17] S. Lawrence, C.L. Giles, K. Bollacker. Digital Libraries and Autonomous Citation Indexing. IEEE Computer, Volume 32, Number 6, pp. 67-71, 1999

[18] T. R. Leek. Information extraction using hidden Markov models. Master's thesis, UC San Diego, 1997

[19] A. McCallum, D. Freitag and F. Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. ICML-2000

[20] I. Muslea. Extraction Patterns for Information Extraction Tasks: A Survey. In Proceedings of Workshop on Machine Learning and Information Extraction (AAAI-99) pag. 1-6, Orlando, Florida, 1999

[21] J. Pearl (1998) Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference, Morgan Kaufmann.

[22] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1-47, 2002

[23] I.H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco, 2000 http://www.cs.waikato.ac.nz/ml/[2]

[24] Y. Yang An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1(1/2), pp 67–88, 1999