

# Automated Ontology Elicitation and Usage

Adrian Silvescu

Ph.D Thesis proposal

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Background & Motivation . . . . .	2
1.2	Proposed Work . . . . .	3
1.3	Validation . . . . .	3
1.4	Statement of Value . . . . .	4
<b>2</b>	<b>Related Work</b>	<b>4</b>
<b>3</b>	<b>Ontology elicitation</b>	<b>6</b>
<b>4</b>	<b>A Probabilistic Graphical Model for Shallow Parsing - Probabilistic Ontology Elicitation</b>	<b>11</b>
<b>5</b>	<b>Research Plan</b>	<b>16</b>
5.1	Overview of Ontology Elicitation and Usage . . . . .	16
5.1.1	Ontology elicitation overview . . . . .	16
5.1.2	Ontology usage . . . . .	19
5.2	Preliminary results and prior objectives . . . . .	19
5.3	Research questions . . . . .	22
5.3.1	Ontology elicitation (Conditional + Unconditional) in the model free case . . . . .	22
5.3.2	Ontology elicitation (Conditional + Unconditional) in the model specific case . . . . .	23
5.3.3	More probabilistic models that use ontologies . . . . .	23
5.3.4	Other topics . . . . .	26
5.4	Validation: Datasets . . . . .	26
<b>6</b>	<b>Timeline</b>	<b>27</b>
<b>7</b>	<b>Conclusion and Anticipated Contributions</b>	<b>27</b>

## Outline

In the first section of this proposal, we introduce the problem of ontology elicitation and some of the motivation for examining it. In section 2 we critically examine some related work. Subsequently, in section 3 we give a more precise problem description and some intuitions about its solution. We present a probabilistic model associated with Ontology Elicitation in section 4, followed by the description of our previous work and the research plan in section 5. Section 6 contains an estimated timeline. We conclude in section 7 with the anticipated contributions.

## 1 Introduction

In section 1.1 we introduce some background and motivation for the ontology elicitation problem. In section 1.2 we outline the proposed work preceded by some definitions. We discuss some ways of validating the feasibility of our approach in section 1.3 and we conclude in section 1.4 with a statement of value.

### 1.1 Background & Motivation

Data Mining, Knowledge Discovery and Machine Learning are aimed at finding automated ways to analyse data, and occasionally knowledge, from an application domain and provide descriptive or predictive models pertaining to the whole or some aspect of that particular domain.

The overwhelming majority of problem settings in these fields rely on the existence of some primitive concepts/features (representing an initial/apriori ontology) in terms of which the problem, including its goals, is phrased and then subsequently solved. The choice of this initial ontology may have a big impact on the tractability of the problem under consideration (e.g., learning the definition of an arch is more difficult to get from raw pixel data, than from images annotated with primitive geometric objects or contours).

In unstructured domains such as text, protein sequences, temporal sequences, images, reinforcement learning tasks, etc, an ontology over the data space is not always readily available or is largely incomplete. The incompleteness problem can appear in structured domains as well (of which relational table data is a typical example) - because in general we can always extract potential benefits from further “structuring” the domain under study.

As a consequence there is a need for automatic methods that can “structure” the aforementioned domains and consequently provide to learning (or decision making) algorithms a useful feature/conceptual/relational representation that will presumably facilitate the accomplishment of various tasks pertaining to these domains. In fact, the lack of good methods for dealing with this problem is one of the main critiques against the feasibility of human level AI, and also a bottleneck for the advancement of the AI in general [20, 31].

## 1.2 Proposed Work

**Definitions:** By *unstructured domains* we mean domains where a lot of atomic features are present in the input data, but this input data is not annotated with high level features (e.g., those domains mentioned above - sequences, [text, protein sequences, etc.], graphs, etc.). By *structured domains*, on the other hand, we mean domains where some high level features/concepts and/or relations are defined over the data space (e.g., as is the case with data typically found in a relational database). By *ontology elicitation* we mean the process of structuring / “making sense” of a certain application domain by positing the existence of certain features / concepts / entities and relations among them, that are relevant to the domain in question. We will get more explicit later as to how to make more concrete this process and its corresponding elements. <sup>1</sup>

### Main objectives of the thesis:

1. The design, analysis and development of automated methods for ontology elicitation (structuring of application domains) / feature / concept / relation construction in topological and non-topological application domains.
2. Development of Usage and Validation procedures in order to evaluate the Quality and make Effective Use of the Elicited Ontologies in several Application Domains [protein sequences, text data, ...] .
3. Developing a framework for ontology elicitation and usage and explore some of its theoretical underpinnings.

## 1.3 Validation

The success of the work, that is the effectiveness of the produced ontologies, would be assessed from at least two points of view

- Improved performance in certain tasks related to the domain under study such as classification, information extraction, etc. [e.g., evaluation of predictive accuracy for these tasks]
- Comprehensibility of the derived ontologies evaluated either in itself (by the means of a complexity measure) or by human means (such as by comparison with human made taxonomies when available).

---

<sup>1</sup>Philosophical note: The use of the term ontology in the following is rather different from the traditional use, namely, concepts / entities and relations among them as they are in the “true reality”. Our ontologies are a weaker notion in the sense that there is no claim about them pertaining to the “true reality” but rather a conjectural set of concepts / entities and relations among them posited by a “learning” process. In this sense they are pertaining to epistemology rather than to ontology / metaphysics. Probably a better name for them instead of ontologies would be that of epilogies (epi comes from epistemology). In general we take the position that the role of Knowledge Discovery / Learning is not that of producing an objective description of the “true reality” as such a thing is not deemed to be possible, but rather to produce increasingly better conjectural descriptions that yield a higher quality approximation of our data / experiences. These description however are not allowed to acquire ontological / metaphysical status in the traditional sense (see also [46]). So from now on whenever we use the term ontology, we really mean epiology.

## 1.4 Statement of Value

Usefulness of the results: Ontologies, as the ones purported to be derived by our method have at least a twofold use: The first one, which is intrinsic, is that they can provide insight into the application domain under study, by structuring it, and therefore allowing for its examination at a more appropriate level of abstraction than the one allowed by the raw atomic features.

The second one, which is extrinsic, is that ontologies can be used in order to facilitate the accomplishment of various tasks within the given application domain (such as classification, prediction, faster reasoning/planning, etc).

## 2 Related Work

In this section, we briefly review work that has some bearing on the proposed work on ontology elicitation. Such work spans several areas of artificial intelligence including:

1. Feature Construction[22, 14]
2. Hierarchy construction by clustering for text classification [3, 13, 41]
3. Aspect Based Clustering [21, 4, 6]
4. Frequent Item Sets and Frequent Item Sets with Abstraction [1, 42, 18]
5. ILP, Predicate invention [48, 26].
6. Meta Learning - a classifier becomes a feature for the MetaLearner[47]
7. Grammatical Inference [36, 43]
8. Kolmogorov Complexity[7]
9. Constructive Neural Networks [35, 12]
10. Constructing Hidden Variables [11]
11. Probabilistic Context Free Grammar.[29]
12. Shimon Edelman's work on patterns for NLP. which correspond to one abstraction in the middle and 1 or 2 superstructures combined in one pattern is in the same spirit for language[10].
13. Hierarchical HMMs [32, 5]
14. Constructivism [9]
15. Graph and Sequence Kernels [28, 16, 17], Graph Grammars [23, 24]
16. NLP , Syntax & semantics...[34] [44]
17. Structure learning [27, 15]

18. Feature induction Conditional Random Fields [30, 8]
19. Boosting Feature induction [33, 45, 8]

Most of this work is aimed at construction or discovery of features or concepts that enrich the representation of the initial instance space. We will examine these techniques from three points of view:

The *first* one regards the generality of the method, or equivalently how representative is the set of concepts that can be derived using the respective method. >From this point of view we have very complex methods such as ILP (5) and Kolmogorov Complexity (8) and some of the techniques from Grammatical Inference (7) and Constructing Hidden Variables(10), which do not scale up very well to large domains. These research directions also contain a good deal of open problems. At the other end of the spectrum we have problems with better tractability but more naive or narrowing assumptions such as (1,2,3,4,9) which focus on more restricted problems. In this second vein we can further divide the field into two main categories: Clustering/Abstraction Techniques (2,3) and Superstructuring Techniques (1,4,9) with a little but notable cross-fertilisation between these two, attempted in a limited setting in [42] (more exactly Superstructuring based on a given set of abstractions as opposed to only on the basic features)

Our techniques, to be presented in the next sections are targeted to take the middle ground between the very general and the very specific techniques, which is fairly open currently. This is to be accomplished by attempting an integration between Abstraction and Superstructuring. In this way we improve expressivity while keeping the computational requirements within tractable limits.

The *second* perspective along which we can divide the work in the area of feature/concept construction is the availability of a well founded quantitative framework that can provide a fine grained evaluation of the quality of posited concepts/features. In this respect some of the general techniques such as ILP or Grammatical inference, are lacking, while some others are not. In those good cases however pretty much all the problems are open or intractable such as in the case of PCFGs, Hidden Variable Discovery and Kolmogorov Complexity. Some of the simpler methods have a well founded framework (2,3,4) for evaluating the quality of the invented concepts but again they are lacking expressive power. Also more notably Frequent Item Sets and Frequent Item Sets with Abstraction which are less principled in dealing with hidden variables than probabilistic approaches and the elicited concepts are special kinds of hidden variables.

We will use probability theory as a quantitative framework in order to solve the credit assignment problem associated with the posited new concepts/features.

The *third* point of view from which the Hidden Variable / Feature Construction research area can be examined is that of search method employed. The most straightforward method is exhaustive enumeration, but this method succumbs very early under the burden of combinatorial explosion in complexity (e.g. Kolmogorov Complexity: Enumerate all the Turing machines in pseudo-lexicographic order until you find the right one). The more sophisticated methods can be divided in two main model construction methodologies: Selectionist and Constructivist.

The *Selectionist* methodology proceeds basically as follows: A set of hidden variables/extra features is postulated in the beginning and then a structure is searched for

in the set possible models that contains these variables. The purpose of the search is to select the structural model which has the highest score from the set of structural models that contain the postulated initial hidden variables - hence the name selectionism. Most of the search strategies in this vein are of the hill-climbing type (or more generally stochastic iterative search) either individual (ala MCMC) or population based (e.g., genetic algorithms).

The *Constructivist* methodology (a.k.a. epigenetic/developmental) on the other hand has an emphasis on gradual evolution of the posited new concepts. One way to to briefly characterise the constructivist approach is “You will discover next only those things that you almost already know”. Based on this description the constructivist approach may not seem terribly impressive but its power consists in recursively repeating the process. A few iterations may be able to lead you quite far from the starting point.

The emphasis of the thesis will be more on constructivist methods.

### 3 Ontology elicitation

**Problem Statement** Given some data from a particular domain under study elicit a set of concepts [0/1 annotations of the data points] and relations among them that are “relevant” to the particular domain under study. The set of concepts and the relations among them represent the ontology that is elicited.

**The main paradigm** We will adopt the following main paradigm in order to guide our attempts to solve the ontology elicitation problem:

*“There are at least two essential moves that people make when trying to make sense of or structure a new application domain: Super-Structuring and Abstraction. Super-Structuring is the process of grouping and subsequently naming a set of entities that occur within “proximity” of each other, into a more complex structural unit (e.g., four wheels and a chassis on top of them will be called a car). Abstraction, on the other hand, establishes that a set of entities belong to the same category, based on a certain notion of “similarity” among these entities, and subsequently names that category (e.g., cows, cats and dogs will be called mammals).” [39]*

Note that in order to make more precise the above stated paradigm we need to define more exactly what we mean by “proximity” and “similarity”.

To define the notion of “proximity” we assume that the data from our application domain has a topology (neighborhood system) imposed over it. A topological space is a space where for each element in the space a set of neighborhoods has been defined. The neighbors in this topological space will be considered proximal entities. For example, in the case of sequence data we can define as neighbors of an entity the entities located to its left and to its right in the sequence. In the case of images we can have the neighborhood given by the four/eight adjacent pixels, or the pixels within a radius of k pixels of the pixel under question. In general, for arbitrary discrete topologies (i.e., graphs [these topologies can occur in relational learning settings for example]) we can define as proximal entities, the entities that are reachable from the entity in question by traversing no more than k edges in the graph (sphere of radius k). By the same token, in the more traditional attribute/value representation of machine learning settings two

attribute values can be defined as within proximity of each other if they co-occur within the same instance. More exactly, for the one table case all attributes of an instance are proximal to each other and we have consequently a complete graph topology within one instance.

We will use a distance measure to specify the similarity between two entities. The distance will be calculated as a function of some features of the two entities between which the distance is computed.

**Structuralism vs. Functionalism** Based on how we choose the features in terms of which we will define our distance measure, we can identify two extreme approaches: Structuralism and Functionalism. Remember that what we are trying to define is a distance that reflects similarity between two entities. This naturally leads us to examine the two major philosophies for defining semantics: Structuralism and Functionalism. Assessing the similarity in meaning between two objects is dependent on how we define what those two objects mean. Structuralism asserts that what an object is and therefore what it means can be defined in terms of, its structural features. For example, in order to define a chair we say that it has four legs, a rectangular surface on top of them and a back (note the analogy with class definitions in Object Oriented Programming). Functionalism on the other hand asserts that the meaning of an object resides with its usage. That is, a chair is something that you can sit on, regardless of its form or shape. And if you intentionally kill somebody with that chair, then the chair is for all practical purposes a weapon. (The equivalent of functionalism in Object Oriented Programming is represented by the notion of Interface which basically specifies a bunch of properties necessary for the object to satisfy in order for it to be able to be fit within certain contexts. For example, a Stack needs to have only isEmpty/Push/Pop/Clear functions in order to be used as a Stack.) Briefly stated the functionalist claim is that meaning is about ends and not about the means by which these ends are met. While Functionalism is presumably more appealing, it raises the question of how to operationalize the intuition behind it, in order to define similarity in meaning. That is, how to operationalize the intuition that two objects are similar in meaning if they are used in the similar ways. For cases where the entities under question occur within certain contexts one way to define similarity in meaning is to say that two entities are similar if they occur in similar contexts. Note that given a topological space and hence a notion of proximity (given by neighborhood-ness) the notion of context can easily be defined as the set of proximal entities. We will return to the problem of defining similarity in meaning in the next section, where we will make it more precise in the case of sequences (such as for words in text and aminoacids in proteins). So far, we have established that if the data is embedded in a topological space we can define similarity between two entities by examining the similarity of proximal elements.

One final remark about Functionalism vs. Structuralism is that these two are extreme positions and that a more convenient middle ground may be found between the two by using both structural and contextual features in order to define meaning and similarity in meaning thereof. In the remaining of this section we will present some support for the main paradigm and then describe a general method for Ontology Elicitation.

---

**Algorithm 1** Ontology Elicitation

---

**until** a terminal criteria has been reached

1. abstractions = *Abstract*(data);
2. super-structures = *SuperStructure*(data)
3. data = *Annotate*(data, [with the] abstractions, [and] super-structures)

**repeat**

---

**Inspirations and intuitive justifications for the main paradigm** [OOP/Software Engineering/UML] Software engineers, Programmers and Computer Scientists in general are faced on a daily basis with the very problem that we are trying to solve here: Ontology Elicitation. The job of a Software Engineer is to take a real world problem, transpose it into the computer world and then solve it over there. In other words, the problem consists of two parts: problem definition and problem solving, or coming up with a question and the answer. In a certain sense ontology elicitation is concerned about coming up with the right question rather than coming up with the right answer as most of the algorithms cookbooks are concerned with.

The transposition of the real world problem into computer terms involves sorting out the application domain into categories and identifying their structural properties. Based on this more in depth knowledge appropriate problem formulations can be made. By examining Object Oriented Programming and Software Engineering / UML literature (e.g., [19],[37]) we observe that there are two main tricks that are used in order to model application domains, namely - compositionality and abstraction in Software Engineering jargon, or class definition by grouping together smaller structural units and inheritance in the OOP jargon. And these two tricks correspond to our two postulated moves. Psychological evidence is also to be investigated.

**The general method for Ontology Elicitation** The ontology elicitation procedure is summarized by the algorithm 4.

In algorithm 4 *Abstract* is a procedure that returns the set of the ks topmost ranked abstractions as defined by the distance that measures similarity (the more similar the entities grouped together by an abstraction, the higher this abstraction is ranked). *SuperStructure* is a procedure that returns the set of the ka topmost ranked superstructures formed out of entities that are close to each other according to the given notion of proximity (the structures can be ranked according to the likelihood of the composing entities occurring together significantly above chance level criteria, which can be easily modeled by an independence test: the more dependent the components of the structure are, the more likely it is that this is a true structure and not an artifact composed out of a bunch of entities occurring within proximity of each other by mere chance).

Note that the two operations SuperStructuring and Abstraction have contrary effects. Namely, Abstraction increases compression but decreases modeling accuracy while SuperStructuring increases modelling accuracy.

**Operationalization of the previous intuition** In order to operationalize the general method presented in the previous section we need to make more precise the notions of similarity, proximity, context and true structure vs. artifact. This will basically enable us to specify the Abstract and SuperStructure procedures.

In the case of sequence data, given a set of sequences (e.g., text sentences or protein sequences) one way to operationalize the above paradigm is by defining: proximity - as occurring on adjacent positions in the sequence or being no farther than  $k$  positions; context - being defined as the entities occurring one position to the left or one position to the right, or within  $k$  positions to the left or right respectively; true structure vs. artifact are defined using an independence test that will be detailed later (the more dependent a bunch of entities are the more likely they are to be a true structure versus an artifact). In order to define similarity in meaning between two entities (e.g., words), we will say that the two entities (words) are similar if they are used similarly, i.e., if they are used in similar contexts. This kind of idea can be traced down to the British Empiricists (most notably Jeremy Bentham) and even Aristotle - and more recently to Zelig Harris (1953) in the philosophy of language, where it resides under the name of the *distributional hypothesis*. We will make this notion of similarity even more precise in the definition of Abstraction outlined below.

Given the previous particular definitions we can proceed to operationalize Abstraction and SuperStructuring as follows:

**SuperStructure(S->AB)** - returns the topmost  $k_s$  structures made out of two components that co-occur within a small distance of each other and are unlikely to occur together by chance. One way to measure this is by an independence test - the more dependent two entities are, the less likely it is that the current structure is an artifact. Ranking the sequences according to this dependence likelihood allows us to order the putative structures in order to be able to pick the topmost. We perform the independence test by measuring the KL divergence between the probability of two components occurring together and the probability of them occurring together as independent events (i.e., the product of the marginal probability distributions) -  $KL(P(AB)||P(A)P(B))$ .

Another possibility is to rank structures according to how frequently they occur or equivalently, how much they would compress the data if the structure name would be used instead of the whole structure, thus leading to an MDL type of approach. Note: Chi-squared can also be used for independence, but the Chi-squared test is only an approximation of the KL divergence.

**Abstraction(S-> A | B)** - returns the topmost  $k_a$  abstractions (clusters) of two entities, ranked according to the distance between the contexts in which the two abstracted entities appear. More exactly, we define the distance between the left contexts of the two entities, as the Jensen-Shannon distance between the probability distribution of entities that occur to the left of each entity. Similarly we define a distance for the right contexts. Subsequently, in order to obtain a distance between two entities we can sum up the distances corresponding to the left and right contexts, respectively. We can even consider two types of abstractions, one for the right context and one for the left one.

**Example** In Algorithm 2 we present a run of our algorithm on a text toy data set in order to illustrate some of the intuitions and to examine the potential power of this

---

**Algorithm 2** Example

---

Step1 data:

Mary loves John.  
Sue loves Curt.  
Mary hates Curt.

Abstractions 1:

A1 -> Mary | Sue - because they have similar right contexts: loves.  
A2 -> John | Curt - because they have similar left contexts: loves.

Step 2 data:

[Mary, A1] loves [John, A2].  
[Sue, A1] loves [Curt, A2].  
[Mary, A1] hates [Curt, A2].

Abstractions 2:

A3 -> loves | hates - because of high similarity between their left and right contexts:

This illustrates how abstraction begets more abstraction (A3 not possible on the raw data).

Step 3 data:

[Mary, A1] [loves, A3] [John, A2].  
[Sue, A1] [loves, A3] [Curt, A2].  
[Mary, A1] [hates, A3][Curt, A2].

Structures 3:

S1 -> A1 A3 because it occurs three times  
S2 -> A3 A2 because it occurs three times

This illustrates how abstraction begets structuring (S1 and S2 are not possible on the raw data)

Structures 4:

S3 -> S1 A2  
S4 -> A1 S2

This illustrates how structuring begets more structuring

---

procedure run multiple times:

Note that the abstractions and super-structures derived at one step beget more abstractions and super-structuring at subsequent steps, which cannot be derived directly, based on the initial raw data. Note also that we allow multiple ontologies / points of view by allowing one entity to be abstracted or super-structured in more than one way (e.g. A3 in S1 and S2), unlike pure compression approaches such as [23, 24]. The multiple ontologies / points of view are a very common thing in real life, e.g., somebody can be both a father and a researcher.

So far, we have showed how we can operationalize the ideas presented in the main paradigm. We will use the structures and abstractions derived by our algorithm in order to help in achieving a task within a given domain under study. Thus, the added value obtained by using these derived ontologies in the process of achieving a certain task will be used as a quantitative measure of the usefulness of the derived ontologies.

---

**Algorithm 3** Generic Ontology Elicitation

---

**repeat**

1. *Propose New Concepts* - Using the **Data**
2. *Evaluate* the desirability of the proposed **New Concepts** according to a **Scoring**

**Function**

3. *Select Posited Concepts* according to the ranking of the **New Concepts** by the **Scoring Function**

4. *Annotate Data* with **Posited Concepts**

**until** termination criterion reached

---

## 4 A Probabilistic Graphical Model for Shallow Parsing - Probabilistic Ontology Elicitation

[Overall description] In order to deal with the Ontology Elicitation problem we are going to use the overall approach described in Algorithm 3.

We will show how this generic approach can be instantiated in a probabilistic framework. The choice of a probabilistic framework comes from the fact that in this case, the likelihood difference between models allows us to evaluate the contribution that each newly derived concept has to the overall model. Thus, in this case the instantiation of the Generic Ontology Elicitation steps is as follows:

1. Proposition - propose SuperStructures in order to gain modeling accuracy and Abstractions in order to reduce the model size and gain generalization accuracy.
2. Evaluation - Use EM to calculate the Expected Likelihood
3. Selection - Pick the best
4. Annotation - A slightly more general version of parsing

We further specialize this technique for the case of sequences. The resulting algorithm is described in what follows under the more concrete name of : A Generative Model for Shallow Parsing Sequences [40].

Although the sequence case may seem quite particular, it will allow us to showcase all the issues pertaining to the more general topological case. Therefore, in this section we will focus on sequences. The general cases where we have non-topological data (single table data) or graph based data (e.g., relational data or images) are only extensions of the sequence case. Note that the non-topological case is a particular case of the topological case, where we have a complete graph topology.

In general, automatic ontology elicitation can be seen as a variant of shallow parsing grammar induction. More precisely, ontology elicitation is about parsing the data, because we have to annotate the data after we come up with a new ontology. And since the ontology is really described by a slightly more general kind of a grammar, ontology elicitation is basically about coming up with this grammar.

Therefore, in what follows we will describe some of the research questions that arise when trying to tackle the problem of Ontology Elicitation in Sequence domains,

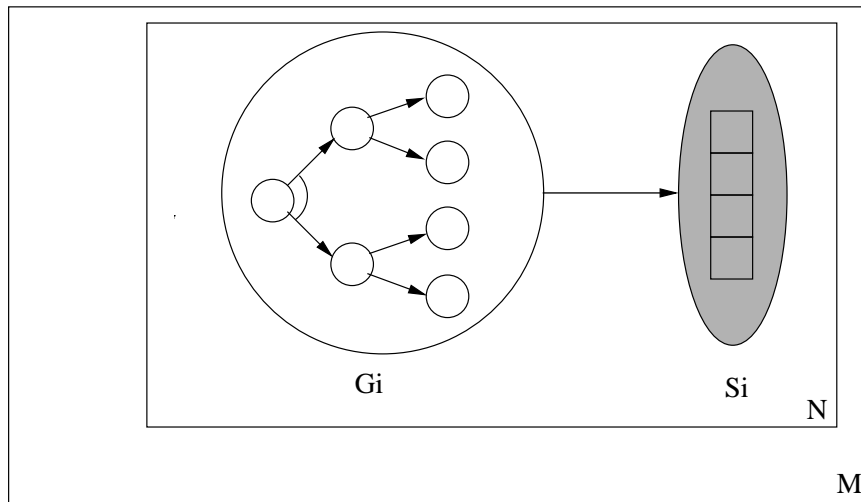


Figure 1: The generative model for shallow parsing

and (as argued previously) without a significant loss of generality. Similar problems appear both in more general topological domains and also in non-topological domains. Thus, we will proceed with the description of the Generative Model for Shallow Parsing for both descriptive and predictive purposes. For predictive purposes (e.g., classification), a generative model is constructed for each class and then the predicted class of a new instance is the one whose generative model assigns the highest probability. Note: More complicated discriminative versions can be considered in order to avoid some of the pitfalls of this approach, but they would not affect the general ideas presented in what follows.

**Introduction** Probabilistic Models for sequence data can be mainly divided into two categories: I. Fairly sophisticated models that are aimed at finding an all encompassing characterisation of the whole sequence by the means of an interdependent generative process (such as PCFGs and HMMs [29]); and II. Relatively simple models that make an independence assumption regarding the generation process of each of the elements of the sequence (such as Unigram, Naive Bayes, Clustering, PLSA and LDA [4]). In the following, we explore the interval between these two extremes with a model that does not attempt to be a global characterisation of the sequence as in the case of PCFG/HMM, but yet it does not assume independence among the generative processes of the subsequent elements in the sequence.

**Model** More exactly, we assume that the generation process produces chunks (subsequences) of elements of variable size, and these subsequences are then concatenated in order to yield the whole sequence. The elements within a chunk are dependent on each other and the dependence is quantified by the hidden macro-variable  $G_i$  denoting the current chunk generator. The chunk generators are chosen independently from a

distribution whose possible values consist of the set of all possible chunk generators. [This model can be further extended to dependencies, either directed or undirected among chunk generators, but for the sake of simplicity we will stick to the independence assumption]. The generative process, using plate notation, is depicted in the figure above. Basically, a generator  $G_i$  is first drawn from a multinomial distribution and then it is used to generate the subsequence (chunk)  $S_i$ . The chunk generators in our case are PCFGs (Probabilistic Context Free Grammars), hence the tree model within the hidden variable denoting the generator  $G_i$ . The PCFGs associated with the chunk generators share the structure and parameters among themselves, the only difference being in probability of activating the start symbol of a certain chunk generator. The generated sequence is a concatenation of independent samples from a mixture of (small!) PCFGs.

**Inference** It is obvious that if we know how the generation process occurred, then there is no mystery about how the sequence was generated. However, typically what we are given is just a sequence, and usually for such a sequence there can be more than one way to generate it. All the possible ways can be compactly stored in a parse chart, which basically achieves compression by representing the possible overlaps in different generation processes only once. One problem that arises in this scenario is the following: given a sequence, what is the probability that a certain non-terminal has generated a subsequence of the given sequence?

Therefore, inference in this model is aimed at determining for each generative model  $G_i$  and for each non-terminal  $N_j$  from the PCFGs, what is the probability that each of these hidden variables has in generating a subsequence of the output sequence.

The resulting algorithm has the flavor of the Inside - Outside algorithm for reasoning in PCFGs but needs to be augmented with one two more messages aside from the traditional  $\alpha$  and  $\beta$ . We call these new messages  $\gamma(1 : t)$  and  $\delta(t : n)$ , and the resulting algorithm is called  $\alpha\beta\gamma\delta$  algorithm.

A Most Probable Explanation (MPE) procedure (a.k.a. Viterbi for Markov models) can be also derived, by replacing the sums with max in the appropriate places in our  $\alpha\beta\gamma\delta$ -algorithm.

The  $\alpha\beta\gamma\delta$  algorithm graphically is illustrated in figure2.

The basic problem is how to calculate the probability of a non-terminal  $N$  being involved in a the derivation of elements  $i$  through  $j$  form the sequence. The derivation in terms of the  $\alpha, \beta, \gamma, \delta$  messages is outlined below.

$$\begin{aligned}
 & P(S \rightarrow s[1 : i - 1]N(i, j)s[j + 1 : n] \rightarrow s[1 : n]) = \\
 = & \sum_{1 \leq u \leq i; j \leq v \leq n;} P(S \rightarrow s[1 : u - 1]G(u, v)s[v + 1 : n]|G(u, v) \rightarrow s[u, i - 1]N(i, j)s[j + 1, v] \rightarrow s[u : v])
 \end{aligned}$$

$$P(G(u, v) \rightarrow s[u, i - 1]N(i, j)s[j + 1, v] \rightarrow s[u : v]) =$$

because of the independence among chunk generators

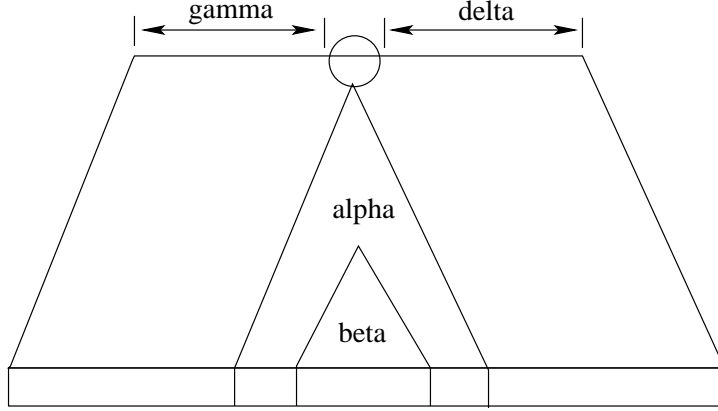


Figure 2: The  $\alpha\beta\gamma\delta$

$$= \sum_{1 \leq u \leq i; j \leq v \leq n;} P(S- > s[1 : u - 1])P(S- > [v + 1 : n])$$

$$P(G(u, v) - > s[u, i - 1]N(i, j)s[j + 1, v] - > s[u : v]) =$$

(by denoting  $\gamma(1 : u - 1) = P(S- > s[1 : u - 1])$  and  $\delta(v + 1 : n) = P(S- > [v + 1 : n])$ )

$$= \sum_{1 \leq u \leq i; j \leq v \leq n;} \gamma(1 : u - 1)\delta(v + 1 : n)P(G(u, v) - > s[u, i - 1]N(i, j)s[j + 1, v] - > s[u : v]) =$$

$$= \sum_{1 \leq u \leq i; j \leq v \leq n;} \gamma(1 : u - 1)\delta(v + 1, n)$$

$$P(G(u, v) - > s[u, i - 1]N(i, j)s[j + 1, v]|N(i, j) - > s[i : j])P(N(i, j) - > s[i : j]) =$$

$$= \sum_{1 \leq u \leq i; j \leq v \leq n;} \gamma(1, u - 1)\delta(v + 1, n)\alpha(u, v, i, j)\beta(i, j)$$

where  $\alpha(u, v, i, j) = P(G(u, v) - > s[u, i - 1]N(i, j)s[j + 1, v]|N(i, j) - > s[i : j])$  and  $\beta(i, j) = P(N(i, j) - > s[i : j])$ .

The Generator hidden variable inference problem is presented next:

$$P(S- > s[1 : u - 1]G(u, v)s[v + 1 : n] - > s[1 : n]) =$$

$$= P(S- > s[1 : u-1]G(u, v)s[v+1 : n]|G(u, v)- > s[u : v])P(G(u, v)- > s[u : v])$$

because of the independence among chunk generators

$$= P(S- > s[1 : u-1])P(S- > [v+1 : n])P(G(u, v)- > s[u : v]) =$$

$$\gamma(1 : u-1)\delta(v+1, n)\beta(u, v)$$

**Learning - Parameters** In order to estimate the parameters of the model we use Expectation Maximization. The Expectation step computes the probabilities outlined in the previous paragraph using the  $\alpha\beta\gamma\delta$ -algorithm. As for Maximization, since our model is decomposable - it has sufficient statistics. Therefore all we have to do in order to compute the maximum likelihood estimator for the parameters is to compute the expected counts by summing up the probabilities obtained during the Expectation step. [Some regularization is also used in the form of Dirichlet priors, but for simplicity we omit the details].

The formula for the expected counts is just

$$EC(N; s[1 : n]) = \sum_{1 \leq i < j \leq n} P(S- > s[1 : i-1]N(i, j)s[j+1 : n]- > s[1 : n])$$

Thus, we have just shown how to calculate  $P(S- > s[1 : i-1]N(i, j)s[j+1 : n]- > s[1 : n])$  used in the inference step outlined above.

[**Note**] Note that even though the  $\alpha\beta\gamma\delta$ -algorithm is conceptually more complicated than the  $\alpha\beta$ -algorithm, we obtain considerable improvements in running time because the  $\alpha\beta$ -algorithm in our case is run on much smaller strings (chunks vs. the whole sequence).

**Learning - Structure** In order to learn the structure of the PCFGs, we use a Structural-EM type of approach with appropriate custom operators designed for this task. We can show that every CFG can be written in and Abstraction Super-structuring Normal Form (ASNF), which restricts the production rules to one of the following types:

1. [Atomic]  $A \rightarrow a$  and this is the only rule that contains  $A$  in the LHS (Left Hand Side) of the rule.
2. [Super-structuring]  $A \rightarrow BC$  and this is the only rule that contains  $A$  in the LHS (Left Hand Side) of the rule.
3. [Abstraction]  $A \rightarrow B$ .

Note that all the uncertainty of this model is associated with abstractions.

As a consequence, our Structural Learning operators will be either of the Abstraction or of the Super-structuring type and with proper randomization, we can show that the procedure is complete. All the Atomics are inserted in the first step by default.

---

**Algorithm 4** Ontology Elicitation

---

repeat

1. *Propose New Concepts* - Using the **Data**
2. *Evaluate* the desirability of the proposed **New Concepts** according to a **Scoring Function**
3. *Select Posited Concepts* according to the ranking of the **New Concepts** by the **Scoring Function**
4. *Annotate Data* with **Posited Concepts**

until a termination criterion reached

---

## 5 Research Plan

This section contains a brief overview of the Ontology Elicitation and Usage framework, followed by an outline of some preliminary results that support the feasibility of our approach. Several research questions that will be addressed are presented. The data sets on which the method is purported to be evaluated are briefly described at the end of the section.

### 5.1 Overview of Ontology Elicitation and Usage

#### 5.1.1 Ontology elicitation overview

As we have seen before, we need to use concepts in order to define and solve tasks in different application domains. However, sometimes the input data is not described in terms of the most appropriate set of concepts that are needed in order to formulate and solve a problem. As a consequence there is an imperious need for methods that can elicit these concepts - i.e., Ontology Elicitation.

The overall strategy for solving the Ontology Elicitation is described in Algorithm ???. Note that this is the same as Algorithm 3, but we repeat it here for convenience.

In order to operationalize this algorithm we need to solve the Proposition, Evaluation, Selection and Annotation problems. One important problem that we need to address, in a general setting, is to provide a scoring function that will allow us to execute the evaluate step number 2 and the select step number 3. In what follows, we will distinguish the scoring functions along two dimensions:

1. Model free vs. Model based
2. Conditional vs. Unconditional

**Model free vs. Model based scoring functions** The scoring functions that can be used to assess the desirability of the New Concepts can be divided into two categories: *model-based* and *model-free*.

The *model-free* scoring functions attempt an a priori evaluation of the quality of posited new concepts without respect to any class of models. An example of such an evaluation strategy is the information gain or KL divergence.

The *model-based* scoring functions evaluate the quality of the posited new concepts with respect to a given model  $m \in M$ . Thus, given a scoring function  $score : M \rightarrow \mathbb{R}$  for evaluating the quality of models in the class of models  $M$ , we will compute the quality of the a newly posited concept  $c$  as:  $quality(c) = score(m \cup \{c\}, Data) - score(m, Data)$ . That is, the quality of a concept  $c$  with respect to a model  $m$  and a data set  $Data$  expresses basically how much more useful the concept  $c$  is aside from the already existing model  $m$  in modelling the  $Data$ . Note that one important requirement for the class of models  $M$  is that they are closed under union with the concepts. (i.e. by adding a concept to a model we obtain a model). In the Probabilistic Model for Shallow Parsing we have presented a class of probabilistic models that are closed under union with concepts.

We have seen that the main topic of this thesis is to inquire how we can come up with new concepts. This inquiry is tightly coupled with the way the concepts are used, the process of forming them being triggered by certain patterns of usage.

Based on the type of usage we distinguish two types of scoring functions and consequently, two types of ontology elicitation procedure:

**Conditional vs. Unconditional** In the *conditional*, a.k.a. predictive case, we evaluate the quality of a posited concept with respect to how predictive it is of a certain goal. A typical model free score function for the class conditional case is the Information Gain about the class conditioned on knowing the concept vs. not knowing it. In the model based case, a typical scoring function would be the difference between the class conditional likelihood/MDL of the data given the model with and without the concept. Note that the real difference between the model free and the model based is the fact that the information gain in the model based is conditioned also on the model.

In the *unconditional* case, on the other hand, the posited concept is evaluated with respect to how well it contributes to modelling the input data. Thus, for example, in the model based case the scoring function is the difference between the likelihood/MDL score of data given the model with and without the concept. Contrast this with the conditional case, where we had the conditional likelihood/MDL score. In the model free case, some examples of scores are the ones presented in section 3. For example, the score for superstructuring two adjacent entities can be the information gained by modelling the joint of A and B versus approximating it with the product of the their marginals. Another possible score for superstructuring is how much compression can be achieved by using one symbol instead of using the adjacent A and B. In the case of abstraction of two entities A and B the score can be conceived in terms of the information lost by substituting A and B with one symbol versus the compression that can be gained this way. Note that minimizing the information lost is equivalent to maximizing the negative information gained by not abstracting vs. abstracting, hence this score is also from the information gain family.

Returning to the overall schema outlined in the Algorithm 3 another issue regards the types of operations that allow to produce new concepts.

**Operators for concept creation: Abstraction, SuperStructuring** So far we have explored two operators for concept creation: Abstraction and Superstructuring. To

recapitulate the intuitions behind these operators formulated in the main paradigm:

1. *Super-Structuring is the process of grouping and subsequently naming a set of entities that occur within “proximity” of each other, into a more complex structural unit (e.g., four wheels and a chassis on top of them will be called car).*
2. *Abstraction, on the other hand, establishes that a set of entities belong to the same category, based on a certain notion of “similarity” among these entities, and subsequently names that category (e.g., cows, cats and dogs will be called mammals)*

One question that can be asked is: should we include more operations and if yes, which ones?

Another question that can be asked with respect to the Ontology Elicitation process is related to the way we intermix the Abstraction with the Superstructuring processes.

**Controlling the Ontology Elicitation process** We distinguish between two cases:

1. Mixing of Abstractions and Superstructuring at each step - base case
2. Use Superstructures in order to improve modeling accuracy and abstractions in order to decrease model size and gain generalization accuracy [i.e., associated with better statistical significance due to larger sample size]

The base case (1) is a potentially quite steep monotonically increasing process, because at each step we introduce a set of  $k_a$  Abstractions and  $k_s$  Superstructures. This can get out of bounds quite quickly depending on how many new concepts per step we allow to be introduced. As a consequence, the second case was devised as a way to implement a memory bounded version of the Abstraction / Superstructuring process, where the number of concepts that are allowed at the end of each iteration is bounded by a number  $M$ . In this case, the process will proceed as follows:

- Alternate between
  1. create Superstructures until the number of concepts reaches the bound  $M$
  2. use Abstractions in order to reduce the current number of concepts to  $\alpha M$  ( $0 < \alpha < 1$ )

Note that the creation of a Superstructure will increase the number of concepts by one (the superstructure). The creation of an Abstraction will also increase the number of concepts by one (the abstraction), but if we eliminate the two abstracted concepts, we get an effective decrease of one concept due to Abstraction. Thus, we can distinguish between Abstraction with elimination and Abstraction without elimination. Obviously, for the case (2) procedure to work non-trivially we need abstraction with elimination. For the base case (1), both types of abstractions can be used.

### 5.1.2 Ontology usage

Moving to the usage of ontologies, by analogy with the model free / model based distinction of Ontology Elicitation, we can distinguish between:

1. Usage of all the derived concepts as a bulk set of augmented features
2. More specialized usage that takes into account the nature of the target classifier / model

We have presented one example from the second category, namely the Probabilistic Model for Shallow Parsing, but this only opens the avenue for addressing the question of whether there are other interesting Probabilistic Models that can incorporate ontologies.

We have reviewed the Ontology Elicitation and Usage processes and classified them according several dimensions. In what follows, we will present some preliminary work and prior accomplished objectives in support of the feasibility of the approach.

## 5.2 Preliminary results and prior objectives

There are several problems that we successfully addressed in the context of Ontology Elicitation and Usage, including:

- (1) A General Framework for Ontology Elicitation and Usage in the model free environment [39]
- (2) A Probabilistic Framework for Ontology Elicitation - Shallow parsing [40]
- (3) Superstructuring & Abstraction for Sequences
- (4) Abstractions + AVT-NBL for attribute-value data [25]
- (5) Decision tree learning in the presence of attribute-value taxonomies: AVT-DTL [50]
- (6) NBk [38, 2]

We will summarize the preliminary results of our approach in what follows. Although we have already described (1) and (2) in the previous sections, we briefly recapitulate them here for completeness:

(1) The general framework for ontology elicitation and usage in the model free environment [39] is composed of two parts, namely the elicitation and usage, respectively. The elicitation part produces the ontology as a set of additional features that annotate the initial data. The augmented data can be given as input to any classifier (or more generally, to a task solver). In this framework, the scoring functions that evaluate the quality of the posited concepts in the ontology elicitation process are designed independently of the particular classifier that will use these concepts, hence the model free nomenclature (one example of such a scoring function is the information gain).

Thus, our framework allows us to create ontologies in a model free setup and then subsequently validate them.

(2) The probabilistic framework for ontology elicitation - shallow parsing [40], on the other hand, is a model specific ontology elicitation and usage framework. More exactly, for the case of sequential data, a probabilistic model that is able to use ontologies is proposed. Structurally, the search for new concepts in this case is similar to the search in the model free case. However, the scoring function in this case is given by the difference between the likelihood/MDL of the input data with respect to the models that contain and do not contain the new concept. In terms of usage, the probabilistic shallow parsing model can be used for classification or as a descriptive model.

(3) Within the framework of (1) experiments have been performed on two datasets: text classification (Reuters) and protein function prediction (SWISSPROT). We used the Naive Bayes classifier for prediction.

By using super-structuring (model free - unconditional) alone we got an improvement of 6% in accuracy in the case of protein function prediction and an improvement of 1.5% in the case of text classification. (Note: the reported results are based on 10 fold cross-validation). More exactly, the data sets that we used are described below:

1. **[Protein Sequences]** We have selected a subset of proteins from the SWISSPROT database in such a way that there are at least 10 instances (proteins) per class, where the class is the protein function as defined by the INTERPRO entry. This subset contains 74000 protein instances (75% of SWISSPROT) and 1641 classes. On this subset we obtained a 6% improvement in accuracy by using SuperStructuring, over the baseline of the Naive Bayes classifier.
2. **[Text Classification]** The data set used for the text classification task was Reuters 21578. This dataset contains approximative 12000 examples and 100 classes. Thus, using SuperStructuring alone, on the task of text classification applied to the Reuters 21578 data set, we obtained an improvement of 1.5% over the baseline of Naive Bayes.

By using abstraction alone the accuracy of the Naive Bayes classifier decreased, due to the fact that abstractions blatantly violate the assumptions of the Naive Bayes classifier. This problem can be fixed by using the abstractions in a slightly more controlled fashion as we did in (4).

(4) Abstractions + AVT-NBL for attribute-value data [25] As opposed to the work in (3), which has been done in a sequential setup (i.e., protein or text sequences), this work is done in a "one table" data setup, where each data instance is represented as a tuple of attributes, which can take values in an attribute values set. There exists a distinctive attribute, called class. In this setup, we have elicited Abstractions over the values of each attribute in the form of a taxonomy tree. This has been done by running a Hierarchical Agglomerative Clustering algorithm, which purports to group together items based on a similarity distance. In our case, the distance between two values of an attribute was given by the similarity distance between the distributions (e.g., KL divergence) over classes that appear in the same instance with each of the two values. (i.e., a model free - conditional setup). Once the taxonomies were generated, we designed a variant of the Naive Bayes classifier (AVT-NBL) that can use these

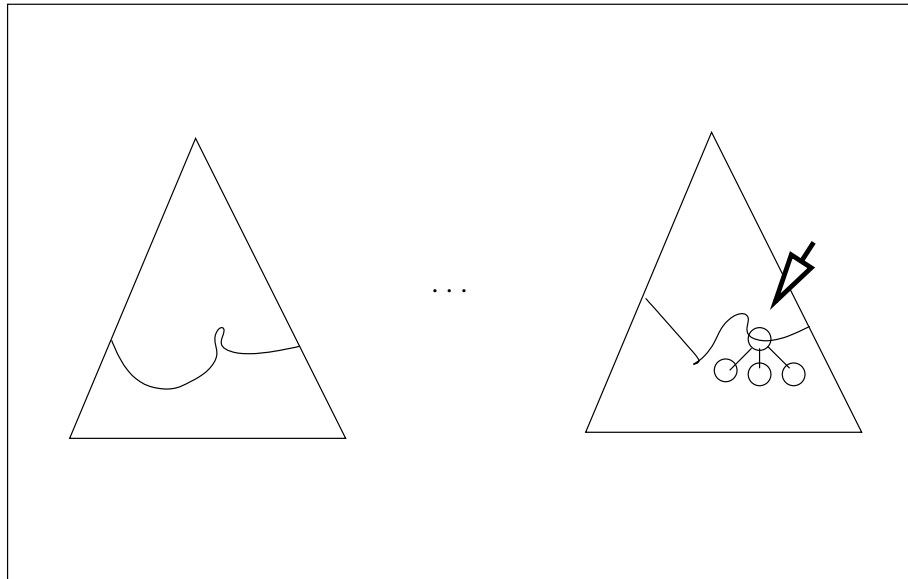


Figure 3: Multiple cuts through taxonomies

taxonomies in the process of learning. The input attributes of AVT-NBL can take values at a higher level of abstraction (compared to the original values), corresponding to a cut through the taxonomy of each attribute. Based on an MDL score, AVT-NBL selects the best such cut. This cut can be obtained by a top down procedure that incrementally expands the current cut to the one that is enlarged with the children of one of the nodes. See Figure 3. Thus, the expansion is pursued in a top down greedy fashion, where the node to be expanded is the one that will yield a classifier with the best MDL score among all possible expansions.

This technique allowed us to improve the performance over the baseline Naive Bayes classifier. The main insight here is that by using a cut through the tree instead of the whole tree, we avoid the problem of introducing a set of highly dependent features. Hence, abstractions can be used to improve classification accuracy, even for Naive Bayes which makes strong independence assumptions.

(5) AVT-DTL [50] In the same vein as the work described in (4), but this time in the context of a Decision Tree classifier, we have shown that we can obtain improvements in classification accuracy by using taxonomies over the values of attributes.

(6) NBk [38, 2] One lesson learned from the experiments performed in (3), (4) and (5) was that abstractions could be used to improve classification accuracy. However, as we have seen, at least in the case of the Naive Bayes classifier, a more carefully designed procedure needs to be employed to achieve that. Thus, even though we have obtained improvements in accuracy in the case of superstructuring, by using the superstructures as a bulk set of augmented features, we can ask the following question: is there a better way of using superstructuring that will result in even better improvements? The answer turns out to be positive. To support this answer we designed NBk

algorithm - Naive Bayes for superstructures that are k-grams.

First, we have derived the most basic superstructuring from a sequence, that is k-grams, which are overlapping subsequences of length k. These subsequences are obtained by sliding a window of length k along the original sequence and advancing one entity at a time. We used the resulting k-grams as features to a Naive Bayes classifier. This gives us the baseline algorithm, which is NB-k-grams. Note that the subsequent overlapping k-grams are quite dependent, since they share k-1 entities. Therefore, this set of features again violates the assumptions of the Naive Bayes classifier. In order to take into account this overlap we have proposed NBk - a probabilistic model that properly discounts the dependencies contained in the overlap. (Note: NBk turns out to be equivalent with class conditional Markov Models). The results of this algorithm compared to the results of the baseline algorithm prove that there are ways to improve the usage of superstructuring with more carefully engineered models.

In what follows, we will examine some more research questions proposed and outline the approach we plan to use in order to answer these questions.

### 5.3 Research questions

#### 5.3.1 Ontology elicitation (Conditional + Unconditional) in the model free case

**Previous work:** The work described in (1), (2) and (4) above addressed some issues related to the problem of ontology elicitation and usage in the model free case (general framework, probabilistic framework, abstractions for attribute-value data, AVT-NBL etc.). We plan to extend this work, as described below.

**Approach & Issues:** As before, we will elicit ontologies based on Abstraction and SuperStructuring in a model free setup. However, we will use SVM as a classifier instead of Naive Bayes, since SVM is a more robust classifier that avoids two major pitfalls that Naive Bayes presents (the strong independence assumption and not maximizing the Conditional Log Likelihood). We will also examine the differences between the two control processes for Abstraction and SuperStructuring: mixing Abstraction and SuperStructuring versus using them sequentially.

**Brief Description.** Ontology elicitation:

- **repeat**
  1. **collect** sufficient statistics from data in order to evaluate the quality of abstractions and superstructuring (these are basically co-occurrence statistics)
  2. **abstract** using a Clustering algorithm (e.g., HAC) based on similarity between contexts of values and produce  $k_a$  abstractions
  3. **superstructure** adjacent entities based on the information gain or compression and produce  $k_s$  super-structures
  4. **annotate** the data with the produced abstractions and superstructures
- **until** a termination criteria has been reached

We will use SVM algorithm to produce a classification on the augmented data.

### 5.3.2 Ontology elicitation (Conditional + Unconditional) in the model specific case

**Previous work:** The work described in (1), (2), (5) and (6) has opened the avenues towards ontology elicitation in the model specific case. We will extend this work as follows:

**Approach:** We will implement the graphical model for shallow parsing and perform ontology elicitation for this model (this is the same as determining the structure of the corresponding grammar). We will study the ontology elicitation control mechanism in this case and also the feasibility of this approach despite its nice theoretical appeal.

**Brief Description:** The implementation of the structure search algorithm is similar to the one outlined in the previous question. The major difference consists of how the scoring function that ranks the quality of abstractions and superstructures is defined. In this case, this function is basically the difference in the likelihood/MDL score of the model with and without the concept whose quality needs to be assessed. The algorithm is outlined below:

- Scoring a concept  $c$  (Abstraction or SuperStructure)
  1. Introduce the production rule associated with the concept in the grammar
  2. Parse the data with the new grammar
  3. Use EM in order to fit the probabilities associated with the model
  4. Calculate the likelihood/MDL score of the data and the fitted model
  5. Return the difference between this likelihood/MDL score of the data and the one without the concept

### 5.3.3 More probabilistic models that use ontologies

**Previous work:** The work described in (2), (4) and (6) deals with probabilistic models that use ontologies in the process of learning. We will extend this work as shown below.

**Approach:** We will investigate how we can produce other probabilistic models that are able to use and incorporate ontologies. We will start with AVT-NBk (or AVT-MMk) and AVT Bayesian Networks, which yield probabilistic models that allow us to incorporate both k-grams and Attribute Value Taxonomies.

We will also investigate how different ontology elicitation methods can influence the final result. In what follows, we present several alternatives to this problem.

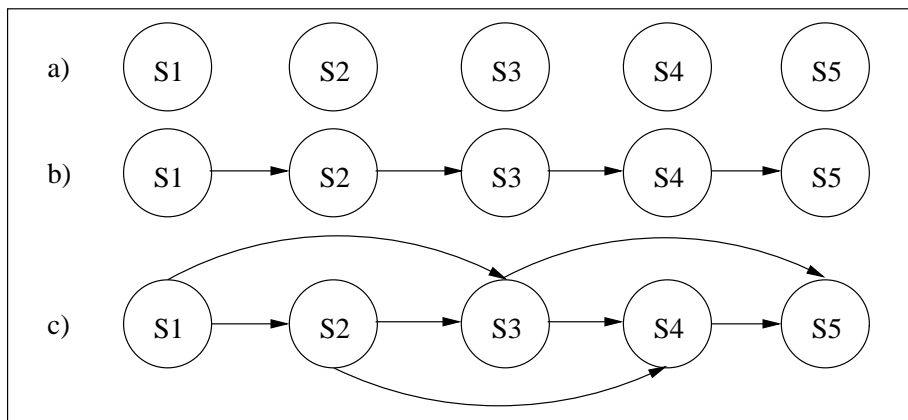


Figure 4: Markov Models of sizes 0-2

**Brief Description (of AVT-NBk/AVT-MMk & AVT-BN):** Markov Models are probabilistic generative models that assume that the probability of an element in the sequence depends only on the previous  $k$  elements. That is, conditioned upon the fact that we know the value of these previous  $k$  elements the next element is independent of the other preceding elements in the sequence. We will call these  $k$  preceding elements the *parent separator*. The Markov( $k$ ) property can be written in quantitative terms as:

$$P(S^t | S^{t-1}, \dots, S^1) = P(S^t | S^{t-1}, \dots, S^{t-k})$$

Some examples of Markov Models of sizes 0 through 2 are shown in Figure 4.

Using Ontology Elicitation we propose to create more concepts in the parent separator space and potentially find a less complicated separator using these concepts. In what follows we describe two simple taxonomy based instantiations of this idea. These two taxonomy based methods have the quality of both being easy to elicit and allowing for an effective structural learning procedure.

**Method I:** Using hierarchical agglomerative clustering we elicit a taxonomy over the values of the elements in the sequence. Thus, for the parent separator  $S^{t-1}, \dots, S^{t-k}$ , we have a cross product of  $k$  identical taxonomies, as can be seen in Figure 5. Then using a top down strategy, we search for the best cut through these set of taxonomies; that will allow us to make  $S^t$  independent of the rest of the sequence given the cut through  $S^{t-1}, \dots, S^{t-k}$  on preceding elements sequence. More exactly, the stopping criteria is an MDL score that balances the amount to which we achieve this independence with the complexity of the model.

**Method II:** Using hierarchical agglomerative clustering we elicit a taxonomy over the values of parent separator  $S^{t-1}, \dots, S^{t-k}$ , viewed as a whole, as shown in Figure 6.

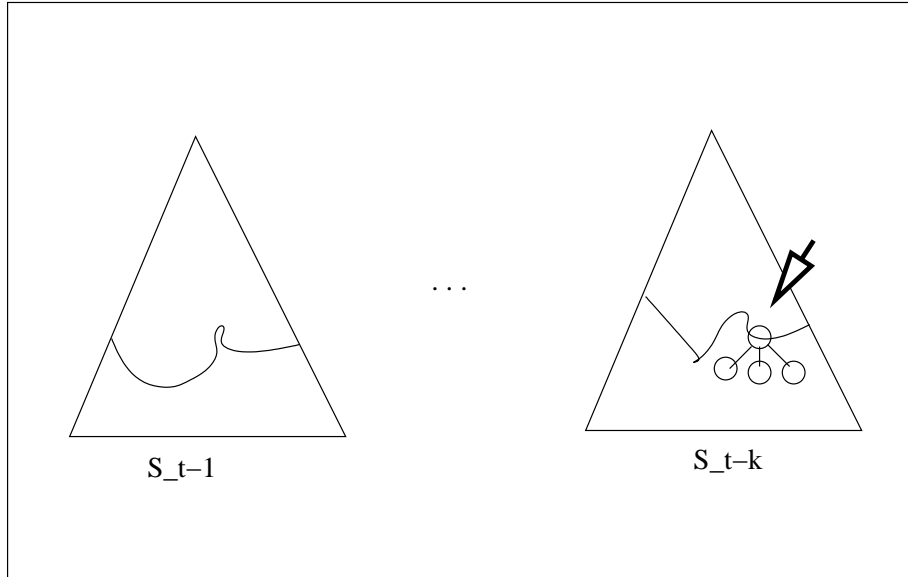


Figure 5: The parent separator taxonomy

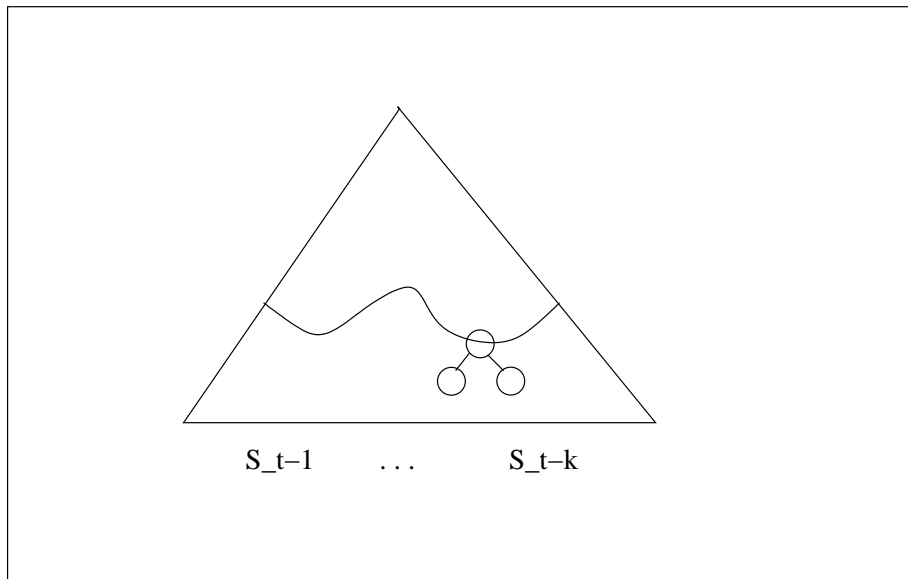


Figure 6: Holistic parent separator taxonomy

Then using again a top down strategy we search for the best cut through this set of taxonomies, a cut that will allow us to make  $S^t$  independent of the rest of the sequence given the cut through  $S^{t-1}, \dots, S^{t-k}$ , but this time viewed as a whole on preceding elements sequence. The stopping criteria, again, is an MDL score instead of mere independence that balances the amount to which we achieve independence with the complexity of the model.

**Note:** Note that if we want to create generative models for data at higher levels of abstraction, then we can also use a top down search over a taxonomy associated with  $S^t$ , in a fashion similar to the one described in [49] and [25]. This is a potential extension that might be worth exploring.

**AVT-Bayesian Networks:** The technique presented above is applicable to arbitrary Bayesian Networks by identifying cuts in the parent separators of each node, cuts that preserve the independence property (or alternatively maximize an MDL score). The identification of the cuts can be viewed as a joint top down strategy over all the parent separators. Note that even though one node can appear in more than a parent separator, the cuts over these parent separators are pursued independently.

#### 5.3.4 Other topics

Other topics that are under investigation include:

1. Relational Learning
2. Regularisation for Probabilistic Classifiers

### 5.4 Validation: Datasets

The datasets that will be used to validate the proposed approach will be selected from the following domains:

1. Text categorization
2. Proteins - functional classification
3. Others

**Text Categorization** The text categorization problem involves predicting the class (usually a topic such as: sports, economical, leisure, etc.) of a document (e.g., news, customer relations, etc.). We are experimenting with the Reuters 21578 data set, which contains approximately 12000 news classified into about 100 classes. Another dataset that we plan to use is the Reuters 1996-1997, which contains 800000 documents classified into about 100 categories.

**Proteins - functional classification** Protein function classification from sequence involves predicting the function (or other attributes such as localization) of a protein given the sequence. We are experimenting with various sequence data sets extracted from SWISSPROT and functionally annotated using GO (the gene ontology). The protein sequences are composed of approximately 200-300 amino-acids ranging in a domain of 20 possible amino-acids. The GO annotation is a functional class for the protein (e.g., Ligase, Isomerase, Helicase, Kinase, etc.).

**Other Datasets** Other datasets may include:

1. Proteins - interaction (collective) classification
2. Intrusion detection sequences

## 6 Timeline

1. [Nov04-Dec04] Experiments with Model free ontology Elicitation
2. [Nov04-Jan05] Experiments with AVT-NBk and AVT-BN
3. [Nov04-Feb05] Experiments with the Probabilistic model for Shallow Parsing
4. [Jan05-Feb05] Write paper on Model free ontology Elicitation
5. [Jan05-Mar05] Write paper on AVT-NBk and AVT-BN
6. [Feb05-Apr05] Write paper on the Probabilistic model for Shallow Parsing
7. [Apr05- Summer05, possibly Fall05] Write thesis
8. [Summer05 / Fall05] Defend

## 7 Conclusion and Anticipated Contributions

Ontologies (posited concepts/entities/relations pertaining to experience) are the means by which we define our very problems. They also allow us to solve these problems better and to produce a more compact description of our experiences.

In this thesis we investigate the following question: What processes can allow computers to postulate ontologies (concepts/entities/relations) based on their experience (data) and how such derived ontologies may be used to aid solving problems in certain application domains.

In this direction we have described: an algorithm for ontology elicitation in a model free scenario; another algorithm in a probabilistic model based scenario; and a way to use ontologies in order to produce more concise version of probabilistic models such as Markov Models (k) and Bayesian Networks.

The ontology elicitation methods proposed in this thesis are of a constructivist flavour as opposed to a selectionist one. In the selectionist case a set of potential ontologies are described and then the job of the learning algorithm is to select the one that

is best supported by the data. In contrast the constructivist approach describes ways to construct ontologies rather than selecting them from a pre-specified set.

Ontologies will be constructed for at least two application domains: text and protein sequences.

In summary the purported contributions of the thesis are:

1. Development of techniques for performing Automated Ontology Elicitation from Data.
2. Development and usage of Validation procedures in order to evaluate the Quality of the Elicited Ontologies in several Application Domains [protein sequences, text data, ...] from two perspectives:
  - (a) accuracy improvement - [with respect to a particular task at hand]
  - (b) comprehensibility - [ e.g., by comparing with human created ontologies or by graph based complexity measures]
3. Developing a framework for ontology elicitation and usage and examine some of its theoretical underpinnings.

## References

- [1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.
- [2] C. Andorf, A. Silvescu, D. Dobbs, and V. Honavar. Sequential probabilistic models for protein function classification. In *KBCS-2004*, 2004.
- [3] Kamal Nigam Andrew McCallum and Lyle Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD-2000*, 2000.
- [4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, Jan 2003.
- [5] H. Bui, D. Phung, and S. Venkatesh. Hierarchical hidden markov models with general state hierarchy. In *AAAI 2004*, 2004.
- [6] Wray Buntine, Sami Perttu, and Henry Tirri. Building and maintaining web taxonomies. In *XML Finland 2002*, 2002.
- [7] Rudi Cilibrasi and Paul Vitanyi. Clustering by compression, 2003.
- [8] T. G. Dietterich, A. Ashenfelder, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *ICML-04*, 2004.

- [9] Gary L. Drescher. *Made Up Minds: Constructivist Approach to Artificial Intelligence*. MIT Press, 1991.
- [10] Shimon Edelman, Zach Solan, David Horn, and Eytan Ruppin. Rich syntax from a raw corpus: Unsupervised does it. In *NIPS-2003 workshop on Syntax, Semantics and Statistics*, 2003.
- [11] Gal Elidan and Nir Friedman. Learning the dimensionality of hidden variables with nir friedman. In *UAI-2001*, 2001.
- [12] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *NIPS-1990*, 1990.
- [13] Naftali Z. Tishby Fernando Pereira and Lillian Lee. Distributional clustering of english words. In *COLING-93*, 1993.
- [14] Peter A. Flach and Nada Lavrac. The role of feature construction in inductive rule learning. In *ICML2000 workshop on Attribute-Value and Relational Learning: crossing the boundaries*, 2000.
- [15] N. Friedman. The bayesian structural em algorithm. In *UAI-1998*, 1998.
- [16] T Gaertner, J Lloyd, and P Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.
- [17] Thomas Gaertner. A survey of kernels for structured data. *SIGKDD Explorations*, 2003.
- [18] B. Goethals. Survey on frequent pattern mining, 2003.
- [19] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification*. Sun Microsystems, 2 edition, 2000.
- [20] John Haugeland, editor. *Mind Design II*. MIT Press, 1991.
- [21] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.
- [22] Hiroshi Motoda Huan Liu, editor. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer, 1998.
- [23] L. B. Holder I. Jonyer and D. J. Cook. Concept formation using graph grammars. In *KDD Workshop on Multi-Relational Data Mining*, 2002.
- [24] I. Jonyer, L.B. Holder, and D.J. Cook. Mdl-based context-free graph grammar induction. In *FLAIRS-2003*, 2003.
- [25] D. Kang, A. Silvescu, J. Zhang, and V. Honavar. Generation of attribute value taxonomies from data and their use in data-driven construction of accurate and compact naive bayes classifiers. In *ICDM'04*, 2004.

- [26] S. Kramer. Predicate invention: A comprehensive view. Technical Report OFAI-TR-95-32, Austrian Research Institute for Artificial Intelligence, 1995.
- [27] W. Lam and F. Bacchus. Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10:269–293, 1994.
- [28] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. Text classification using string kernels. In *NIPS*, pages 563–569, 2000.
- [29] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, May 1999.
- [30] Andrew McCallum. Efficiently inducing features of conditional random fields. In *UAI-03*, 2003.
- [31] Marvin Minsky. *The Society of Mind*. Simon & Schuster, 1988.
- [32] Kevin Murphy and Mark Paskin. Linear time inference in hierarchical hmms. In *NIPS '01*, 2001.
- [33] Kevin Murphy, Antonio Torralba, and William Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *NIPS'03*, 2003.
- [34] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *KDD-2002*, 2002.
- [35] Yang J. Parekh, R. and V. Honavar. Constructive neural network learning algorithms for multi-category pattern classification. *IEEE Transactions on Neural Networks*, 2000.
- [36] Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185:15–45, 1997.
- [37] Joseph Schmuller. *Sams Teach Yourself UML in 24 Hours*. 1999.
- [38] A. Silvescu, C. Andorf, D. Dobbs, and V. Honavar. Inter-element dependency models for sequence classification, [www.cs.iastate.edu/silvescu/papers/nbctr/nbctr.ps](http://www.cs.iastate.edu/silvescu/papers/nbctr/nbctr.ps). Technical report, Department of Computer Science, Iowa State University, 2004.
- [39] Adrian Silvescu and Vasant Honavar. Ontology elicitation: Structural abstraction = structuring + abstraction + multiple ontologies. In *Snowbird-2003*, 2003.
- [40] Adrian Silvescu and Vasant Honavar. A graphical model for shallow parsing sequences. In *ATEM-2004: The AAAI-04 Workshop on Adaptive Text Extraction and Mining*, 2004.
- [41] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Proc. of NIPS-12, 1999*, pages 617–623. MIT Press, 2000.

- [42] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. *Future Generation Computer Systems*, 13(2–3):161–180, 1997.
- [43] Andreas Stolcke and Stephen Omohundro. Inducing probabilistic grammars by bayesian model merging. In *International Conference on Grammatical Inference*, pages 999–999, unknown, 1994. unknown.
- [44] Justin Harris Tim Oates, Tom Armstrong and Mark Nejman. On the relationship between lexical semantics and syntax for the inference of context-free grammars. In *Proceedings of the 19th National Conference on Artificial Intelligence*, 2004.
- [45] Antonio Torralba, Kevin Murphy, and William Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR'04*, 2004.
- [46] Ernst von Glasersfeld. An introduction to radical constructivism. In P. Watzlawick, editor, *The Invented Reality*. New York: Norton, 1984.
- [47] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.
- [48] Stefan Wrobel. *Concept Formation and Knowledge Revision*. Kluwer, 1994.
- [49] J. Zhang and V Honavar. Learning concise and accurate naïve bayes classifiers from attribute value taxonomies and data. In *ICDM '04*, 2004.
- [50] J. Zhang, A. Silvescu, and V. Honavar. Ontology-driven induction of decision trees at multiple levels of abstraction. In *SARA-02*, 2002.