Introducing the UpWrite Predictor

Jason L. Hutchens

***Abstract*— In this paper we introduce the notion of the UpWrite, a generalised process for the automatic construction of hierarchical representations of data developed by Michael Alder as an alternative to King Sun Fu's programme of syntactic pattern recognition, and show how it might be applied to stochastic language modelling.**

Keywords— **Syntactic pattern recognition, stochastic language modelling, syntactic category acquisition, chunking, information theory, prediction.**

I. Introduction

The *UpWrite* is a process for constructing a hierarchical representation of data. It works by iteratively extracting higher-level structure from the representation of the data at one level of the hierarchy in order to determine how the data should be represented at the next level of the hierarchy. Similarities between local models of the data at one level of representation are used to extract this higher-level structure.

The basic goal of the UpWrite is to progressively abstract the data in such a way as to preserve its necessary features, with the intention of using the abstracted representation in applications such as image recognition.

Consider, for example, the image shown in figure 1. This image is a low-resolution bitmap of a triangle, and the lowest-level representation of such an image, as far as a computer program is concerned, is as a two-dimensional matrix of binary values, corresponding to the black and white pixels in the bitmap. This is the level of representation at which any algorithm which looks for objects such as triangles in images must begin.

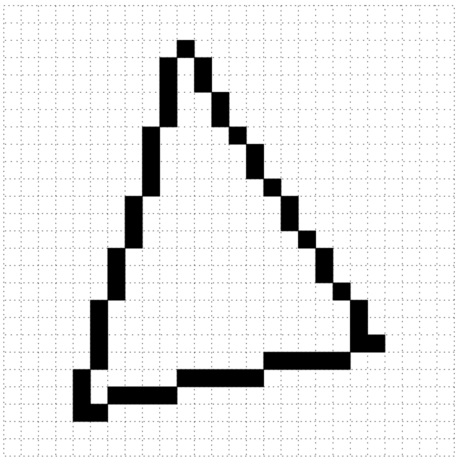

Fig. 1. An binary image of a triangle has several levels of representation, the lowest level of which is an array of bits.

To a human being, however, other levels of representation are immediately apparent. We might say that the next level of abstraction is that of line segments. We can recognise three line segments in the image, and we may additionally recognise three vertices; points which exist solely based upon the relationship of these line segments with one another. This first level of representation, from a collection of pixels to a collection of line segments and vertices, has abstracted the image by grouping together pixels which are, in some sense, parts of the same higher-level object.

At the next level of abstraction, a human being may say that the three line segments and the three vertices describe a single object—the triangle. A computer program which is able to construct a hierarchical model of a triangle in the same way may be capable of determining whether or not a novel image belongs to the set of images of triangles.

The UpWrite can be used to develop a computer program which is able to achieve this classification with a high level of success, without knowing what a triangle is *a priori.* The resulting program is generic enough to be extended to non-trivial image classification problems.

In this paper we shall present the UpWrite by working through a simple image recognition problem, before proceeding to show how the method may be applied to stochastic language modelling. We begin, however, with a few necessary definitions.

J. Hutchens is with Artificial Intelligence. E-mail: hutch@a-i.com

II. Definitions

Definition 1: An *alphabet* is a finite set containing at least two distinct elements known as the *symbols.* Let $\mathcal{A}$ represent the alphabet, let $x \in \mathcal{A}$ denote some symbol from the alphabet, and let $|\mathcal{A}|$ denote the cardinality of the alphabet.

Definition 2: Let $s_z = x_1, \ldots, x_i, \ldots, x_z$, with $x_i \in \mathcal{A}\ \forall\ x_i$, be a sequence of z symbols which we shall refer to as a *symbolic time series*, the *data*, or a *corpus.*

Definition 3: A *stochastic language model* is a model which is capable of assigning a probability $P(s_z)$ to an arbitrary symbolic time series $s_z = x_1, x_2, \ldots, x_z$.

Remark 1: It is common to decompose the probability $P(s_z)$ using Bayes' rule, as in equation 1, where $s_{i-1} = x_1, x_2, \ldots, x_{i-1}$ is the *history.* It should be noted that this decomposition is not unique—there are many possible decompositions, all of them equally valid.

$$P(s_z) = \prod_{i=1}^{z} P(x_i|s_{i-1}) \quad (1)$$

Definition 4: A *predictive model* $\mathcal{M}$ is a stochastic language model which is capable of making a prediction about the next symbol x_i in the symbolic time series s_z in the form of a probability distribution over the alphabet. We denote the probability which the predictive model assigns to the next symbol in the data as $P(x_i|\mathcal{M}, s_{i-1})$.

Remark 2: The inference problem for predictive models is therefore one of estimating the conditional probabilities of the right-hand side of equation 1. This is difficult as i increases, as it becomes more and more likely that the history s_{i-1} will not have been observed in the training corpus at all. The common solution to this problem is to group histories into a finite set of *equivalence classes.*

Definition 5: An *equivalence class of strings* is a set of strings which are deemed to be similar according to some measure. We use $\Phi(s_{i-1})$ to denote the equivalence class to which the string s_{i-1} belongs. The decomposition of equation 1 may now be expressed as in equation 2. Our problem is now one of selecting an appropriate equivalence classification.

$$P(s_z) \approx \prod_{i=1}^{z} P(x_i|\Phi(s_{i-1})) \tag{2}$$

Remark 3: The equivalence classification traditionally used is to classify histories according to their most recent context of n symbols, as in equation 3. This is the *Markovian assumption*, and predictive models which use this equivalence classification are known as n^{th}*–order Markov models.*

$$\Phi(s_{i-1}) = \langle x_{i-n}, \ldots, x_{i-1} \rangle \tag{3}$$

Remark 4: The n^{th}–order Markov model $\mathcal{M}$ estimates the probability of the data s_z as in equation 4.

$$P(s_z|\mathcal{M}) \approx \prod_{i=1}^{z} P(x_i|\langle x_{i-n}, \ldots, x_{i-1} \rangle) \tag{4}$$

Remark 5: The maximum-likelihood estimate of the probabilities of the right-hand side of equation 4 are derived from the normalised frequency with which the substring $x_{i-n}, \ldots, x_i$ occurs in the training corpus. This is achieved as in equation 5, where $C(s_i)$ denotes a count of the number of occurrences of the substring s_i in the training corpus.

$$P(x_i|\langle x_{i-n}, \ldots, x_{i-1} \rangle) \approx \frac{C(x_{i-n}, \ldots, x_i)}{C(x_{i-n}, \ldots, x_{i-1})} \tag{5}$$

Definition 6: The *information supplied to the predictive model by the next symbol in the data* is denoted $I(x_i|\mathcal{M}, s_{i-1})$, where x_i is the symbol which follows the history s_{i-1} and $\mathcal{M}$ is the predictive model, and is given by the negative logarithm, taken to base 2, of the probability of the symbol x_i following the history s_{i-1} according to the predictive model $\mathcal{M}$, that is,

$$I(x_i|\mathcal{M}, s_{i-1}) = -\log_2 P(x_i|\mathcal{M}, s_{i-1}) \tag{6}$$

Remark 6: The information supplied to the predictive model by the next symbol in the data specifies the minimum number of bits required to describe the symbol with respect to the model in an unambiguous way, and may be understood informally to represent the *surprise* the predictive model receives upon discovering what the next symbol in the data actually is.

Definition 7: The *instantaneous entropy of the predictive model* is given by the expected value of $I(x_i|\mathcal{M}, s_{i-1})$, and is denoted $H(\mathcal{M}, s_{i-1})$, where s_{i-1} is the history and $\mathcal{M}$ is the predictive model, that is,

$$H(\mathcal{M}, s_{i-1}) = \sum_{x_i \in \mathcal{A}} P(x_i|\mathcal{M}, s_{i-1}) I(x_i|\mathcal{M}, s_{i-1}) \tag{7}$$

Remark 7: We intuitively think of $H(\mathcal{M}, s_{i-1})$ as the *uncertainty* the predictive model $\mathcal{M}$ has about the next symbol in the data, given that s_{i-1} is the history.

III. History of the UpWrite

In the 1970's Michael Alder became interested in modelling cognitive processes in ways which are consistent with our knowledge of the central nervous system and the findings of Cognitive Psychology. His primary interest was to develop a system which was capable of automatically selecting salient features which would enable a hierarchical representation of the structure in images to be created, thus avoiding the problem in traditional syntactic pattern recognition in which a human being is required to select the alphabet of primitive symbols.

Alder was motivated by the quasi-linguistic features of information processing in the human brain, and a discontent with the classical neural network model. The work of Hubel and Wiesel and Blakemore, described in David Marr's excellent "Vision" [1], suggested that primitive feature extraction occurs at a low level of the nervous system, and this led Alder to develop a framework referred to as the UpWrite.

In the early 1990's Alder, together with Christopher deSilva and Yianni Attikiouzel, authored a series of technical reports describing an early version of the UpWrite [2], [3], [4], and in [3] they write that

> . . . what we seek is not simply a system which can recognise, say, hand-printed or cursive characters, nor one which uses high level descriptions of the characters, but one which generates high level descriptions automatically. Ideally, it might generate a decomposition of the letter A which would amount to recognising three strokes with appropriate relations between them; even more idealistically it might go on to find words and hence to be able to 'read' badly distorted characters simply by using context.

The overall aim of the UpWrite is the formation of a satisfactory theory of learning which may be applied to various problems. Alder, deSilva and Attikiouzel summarised the hierarchical nature of the UpWrite in [4], where they write that

> ... human beings use many levels of description of events and entities. For example, one may specify, in principle, the sequence of hand movements over a piece of paper, or one may say "he signed a cheque". These may be alternative descriptions, of use to different people ... parallel to the levels of description, there are levels of acquisition ...

One of the main features of the UpWrite is that instead of selecting the grammar which is most suited to the current pattern recognition problem, with the result that the selected grammar may be difficult to automatically infer from data, a simple local model which is conducive to inference is selected instead, and techniques are then used to find higher-level structure in the data with respect to the simple local model itself, bootstrapping the model to a higher level of abstraction. The implication of this approach is that identical pattern recognition techniques may be applied to problems which initially appear to be quite different in nature.

The UpWrite has been successfully applied to a wide range of image processing tasks with consistently good performance [5], [6], [7], [8], [9], [10], [11], [12]. This suggests that the UpWrite is an appropriate framework for syntactic pattern recognition, in a more general setting than that prescribed by Fu.

IV. The UpWrite Process

The UpWrite is an attempt to address the problems of syntactic pattern recognition while avoiding the pitfalls of Fu's programme. Specifically, the UpWrite generalises beyond string languages, as they are thought to be too restrictive for the modelling of images, while avoiding the introduction of the human brain into the loop. The UpWrite represents an object hierarchically, at different levels of abstraction, and defines an automatic inference procedure which is used to generate higher levels of description in a process which is essentially bottom-up [13].

Two major types of structure are presumed by the UpWrite: *Sub-objects* are ordered sets of objects, while *quotient-objects* are equivalence classes of objects.

A. The Sub-Object UpWrite

The *Sub-Object UpWrite* is motivated by the fact that data may be partitioned into a collection of sub-objects such that these sub-objects correspond to the primitives of some higher-level representation of the data. For example, a set of pixels in an image may correspond to a line segment, while a set of characters in natural language text may correspond to a word.

Definition 8: The *Sub-Object UpWrite* is a mapping $\langle x_1, \ldots, x_j \rangle \mapsto y$ between a symbol sequence $\langle x_1, \ldots, x_j \rangle$ of symbols taken from some alphabet $\mathcal{A}_1$ to a symbol y in some new alphabet $\mathcal{A}_2$ such that the observed data $s_z = x_1, \ldots, x_z$, $x_i \in \mathcal{A}_1$ can be represented as a higher-level symbolic time series $t_k = y_1, \ldots, y_k$, $y_i \in \mathcal{A}_2$, $k \leq z$.

Definition 9: The *Sub-Object DownWrite* is the corresponding inverse relation.

Incorporating sub-objects into a predictive model has the advantage of increasing the context available to the model, because a single sub-object represents a sequence of lower-level symbols. For example, consider figure 2, which shows a sequence of characters UpWritten to a sequence of character pairs.

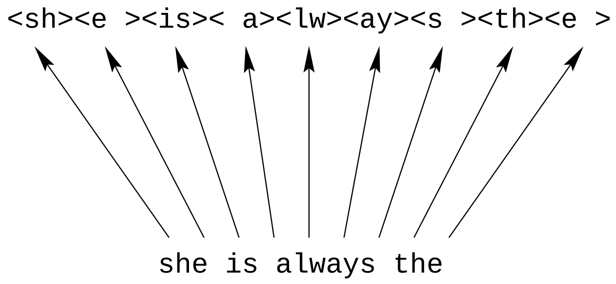

Fig. 2. The Sub-Object UpWrite extends the context available to a predictive model.

A sub-object may be found by observing that a particular set of objects are strongly correlated with one another. Obviously some sort of model is needed to measure the correlation between objects, and the UpWrite requires that a local model of the data be used. This requirement is made partially due to the fact that it enables parallel processing of the data, as many processing units can model different portions of the data independently, partially due to the fact that such models are typically easy to infer from data, and partially due to the fact that the UpWrite itself will extend the reach of each of these local models by constructing a hierarchical representation of the data.

B. The Quotient-Object UpWrite

The *Quotient-Object UpWrite* is motivated by the fact that symbols may be assigned to equivalence classes such that these equivalence classes correspond to the primitives of some higher-level representation of the data. For example, a quotient-object of "things" in an image may correspond to the class of straight line segments, while a quotient object of words in natural language text may correspond to the class of verbs.

Definition 10: The *Quotient-Object UpWrite* is a mapping $\{x_1, \ldots, x_j\} \mapsto y$ between an equivalence class of symbols $\{x_1, \ldots, x_j\}$ taken from some alphabet $\mathcal{A}_1$ to a symbol y of some new alphabet $\mathcal{A}_2$ such that the observed data $s_z = x_1, \ldots, x_z$, $x_i \in \mathcal{A}_1$ can be represented as a higher-level symbolic time series $t_z = y_1, \ldots, y_z$, $y_i \in \mathcal{A}_2$.

Remark 8: The *Quotient-Object DownWrite* is not unique, since it is not possible to determine which lower-level symbol x should serve as the DownWritten version of the symbol y. It is possible to generate possible DownWritten versions of t_z by selecting a symbol $x_i \in \{x_1, \ldots, x_j\}$ at random. The corresponding DownWritten version of t_z, which is not unique, has the property that its UpWrite is

t_z, and the DownWrite therefore provides an indication of how well the UpWrite has captured salient features of the data being modelled.

Incorporating quotient-objects into a model has the advantage of making the most out of the data available, as a single context of quotient-objects corresponds to many contexts of lower-level objects, and therefore will be observed more frequently in the data. For example, consider figure 3, which shows a string of words UpWritten to a string of word classes, where each class contains two words.

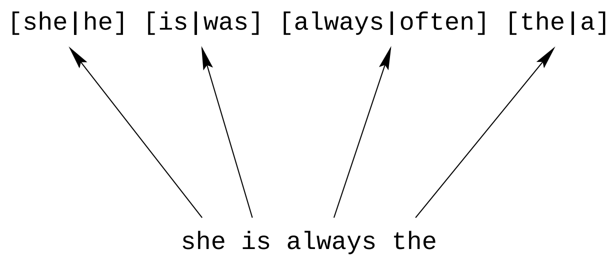

Fig. 3. The Quotient-Object UpWrite results in contexts which are observed more frequently.

A quotient-object may be found by observing that one object may be replaced with another in a particular context without adversely affecting the power of the local model to describe the data. The two objects may then be considered to be equivalent in that the performance of the local model upon encountering either of the objects is similar.

V. Image Recognition via the UpWrite

We shall now give an example of the UpWrite in action. This example is based on the work of Robert McLaughlin and Michael Alder [13], [14], [15], [16], [17], and is purely speculative in that it is not the result of an actual computational implementation.

Consider a pattern recognition system which is required to determine whether a particular image of a polygon is a triangle or a square. To make its job easy, we assume that the images are binary, that exactly one polygon appears in each image, and that there are no noise effects apart from those due to the discretization process.

Clearly this pattern recognition problem borders on the trivial. However, we shall demonstrate that the UpWrite solves it elegantly, and avoids the use of heuristics entirely, meaning that the solution is general enough to be applied to quite different pattern recognition problems which are considerably more complicated.

In order to solve this problem using the UpWrite, we require

- a model which describes the data locally at various levels of abstraction;
- a process for finding line segments and vertices from the local models of the pixels in an image;
- a process for finding squares and triangles from the local models of line segments and vertices; and
- a classification scheme which is capable of deciding whether a hitherto unseen polygon is a square, a triangle or something else entirely.

The reader may wish to pause and consider how a problem such as this would be solved traditionally. It is very likely that a traditional solution would be specifically tailored to look for lines, triangles and squares in images, and would therefore be fairly specific to this particular problem. The usual method employed by scientists and engineers when faced with a problem such as this is to inspect the images by eye, and decide, via introspection, which features of the image allow for their classification, to encode these features in some manner, perhaps by rules and tests, and finally to apply a standard technique for classification. This precludes automation, and restricts the generality of the approach.

It will be shown that the UpWrite is able to find lines, triangles and squares in a very general sense, and does so without human intervention. The UpWrite therefore suffers fewer restrictions than more traditional approaches to pattern recognition.

A. Selecting a Local Model

The UpWrite is a process which represents data at different levels of abstraction, and which does so automatically. This is achieved by modelling the data locally, at each level of representation, and by combining these local models in various ways to find higher-level structure, a bootstrapping process which uncovers global structure at its highest level.

Consider the image of figure 4. We shall assume that this image is expressed as a bitmap of black and white pixels, as it was in figure 1. The lowest-level description of the triangle is therefore a two-dimensional array of symbols taken from the alphabet $\{0, 1\}$, where 0 represents a white pixel and 1 represents a black pixel. For classification of various images to be possible, we need to abstract this data in a way which decreases its degrees of freedom while preserving its structure.

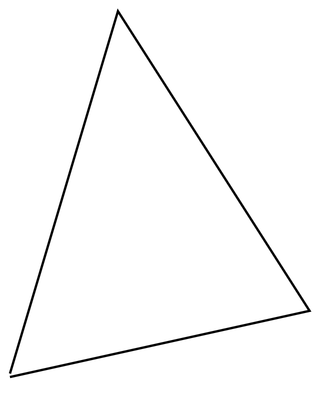

Fig. 4. A line drawing of the triangle of figure 1.

In order to extract higher-level structure from this data, we require a model which is capable of describing a local region of pixels in a way which allows us to measure the correlation between two adjacent regions in order to decide whether they are part of the same higher-level object. We would like to measure this correlation in an information theoretic way; by predicting what we expect to see nearby a particular neighbourhood of pixels. Local *Gaussian Models* possess these necessary features. Although the choice of Local Gaussian Models is not ideal, we do not know of a better alternative at the moment.

B. Local Gaussian Modelling

A local *Gaussian Model* abstracts the data by calculating the normalized 1^{st}– and 2^{nd}–order moments for local neighbourhoods of pixels. This is equivalent to considering the black pixels in the image to be vectors in $\mathbb{R}^2$, and calculating the centroid and covariance matrix of a local neighbourhood of these vectors. The resulting local model, which may be represented as a vector in the UpWrite space $\mathbb{R}^5$, may be shown graphically as an ellipse positioned at the centroid, with an orientation determined by the major eigenvector of the covariance matrix, and a width and length determined by the two eigenvalues.

Figure 5 shows the result of using local Gaussian Models to describe the data. The triangle has been UpWritten from a lower-level representation of 55 vectors in $\mathbb{R}^2$, corresponding to the black pixels of figure 1, to 16 vectors in $\mathbb{R}^5$.

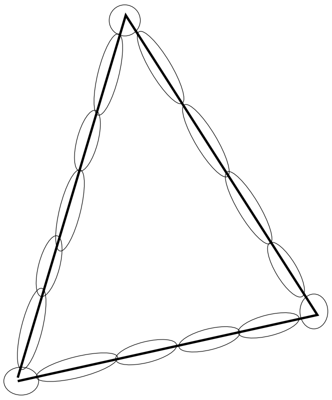

Fig. 5. A line drawing of a triangle with local Gaussian Models superimposed.

This first level of UpWrite has compressed the representation of the triangle by using 80 numbers to describe it instead of 110 numbers. It has also uncovered some structure—we observe that the ellipses positioned along line segments tend to be long and thin, while those positioned at the vertices tend to be quite circular. This first level of UpWrite is an example of the Sub-Object UpWrite, as we have mapped a set of pixels into a single higher-level Local Gaussian Model.

Gaussian Models may be used as predictors. We can, for example, calculate the probability of a particular symbol being black or white using a Gaussian Model, and this enables us to DownWrite the image by colouring each pixel a shade of grey according to the probability assigned to it by the model. This process, if applied to the current model, would produce a triangular fuzz. DownWriting the higher-level representation of the data allows us to confirm that the structure of the data which we consider to be important has been preserved.

C. Finding Lines and Vertices

The triangle is now represented by 16 vectors in $\mathbb{R}^5$. A second level of UpWrite may be performed in order to carry this abstract representation of the triangle further.

As we previously observed, ellipses which lie along line segments tend to be long and thin. If they lie on the same line segment, they have similar orientations, and they are spatially nearby. We can therefore find line segments in the data in at least two different ways.

Cluster the vectors: A clustering algorithm may be applied to find clusters of vectors which are similar according to some metric. Vectors which are assigned to the same cluster may turn out to belong to the same line segment.

Entropic chunking: A sequence of local Gaussian Models may be used to predict what the next model in the sequence should look like. If this prediction proves to be erroneous, we have reached the end of some region of higher-level structure. Until then, we join the sequence of models together. This approach may find line segments, and even curved lines depending on the tolerance threshold used to decide whether a prediction is erroneous or not.

Figure 6 shows the results of this second level of UpWrite, which has successfully identified the three line segments in the image. This second stage of UpWrite is another example of the Sub-Object UpWrite; we have mapped a sequence of local models into a single Gaussian Model at a higher level of abstraction.

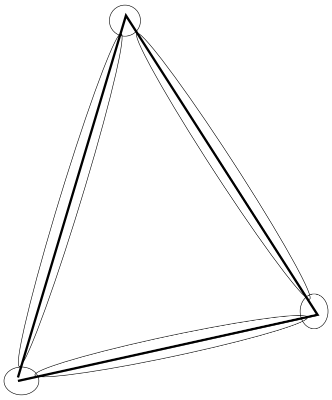

Fig. 6. A line drawing of a triangle with local Gaussian Models describing the lines and vertices.

The vertices of the triangle, represented by circular ellipses in the figure, were not UpWritten at all at this stage of the process. The vertices are areas of high entropy in the image, and may be considered to be separators between line segments.

D. Finding Triangles and Squares

In order to determine whether a set of vectors in high-dimensional space represents a triangle, a square, or something else entirely, we need to perform one further level of UpWrite; the Quotient-Object UpWrite.

We might intuitively describe a triangle as "three line segments and three vertices" and a square as "four line segments and four vertices". We would therefore like our algorithm to automatically decide that all line segments are the same kind of thing, and all vertices are the same kind of thing. If we are successful in doing this, we will have come full-circle, once again describing our data using a binary alphabet!

Contextual information may be used to classify the vectors which represent line segments and vertices into one of these two classes. We know, for example, that vectors representing vertices tend to occur at either end of a line segment, and this information may be enough to provide a classification. We shall not bother with exactly how the classification process is achieved; we shall simply assume that it has been achieved, and that we are left with a single vector in some high-dimensional space which describes the triangle in a very abstract way.

E. Classification

If we apply this UpWrite process to lots of different images of triangles, and lots of different images of squares, and if the UpWrite process preserves enough structure about these shapes so that they may be distinguished at the highest level of abstraction, then we will find clusters of vectors in this high-dimensional space. Classification is then a trivial process of deciding, via some measure of similarity, which of the two clusters a novel vector belongs to, if any. Interestingly, high-dimensional Local Gaussian Models may be used to achieve this classification! Our problem of distinguishing triangles from squares is solved.

In the process of solving this trivial pattern recognition problem, we have represented each image at four levels of abstraction. We therefore have quite powerful models of our images which may be applied to many different problems, such as deciding whether a particular shape is the same as one we have already seen but at a different scale, for example. We also have a pattern recognition technique which can classify a new polygon merely by training the system on a collection of these shapes too, and ensuring that the highest level of abstraction is specific enough to allow it to be distinguished satisfactorily from triangles and squares. If not, we would need to change the representation by adding additional constraints.

VI. The UpWrite Predictor

Application of the UpWrite concept to stochastic language modelling results in the UpWrite Predictor, shown in figure 7, consisting of a chain of modules, each of which contains an UpWriter, for discovering structure in the data using the predictions made by the model in the previous module, an alphabet, for containing new symbols which correspond to the higher-level structure discovered, and a predictive model, inferred from the UpWritten data.[1] Note that predictions made by the model feed back into the UpWriter, allowing erroneous generalisations to be corrected. The power of the technique is the fact that the data is represented, automatically, at various granularities, and that applications may select a level of representation appropriate to the task at hand.

Construction of the UpWrite Predictor may proceed by specifying techniques for

- finding Sub-Objects and Quotient-Objects in the data from the model's representation of it;
- forming a new representation of the data, with respect to this newly discovered structure; and
- correcting erroneous generalisations.

Sub-Objects and Quotient-Objects may be found in data by applying information theoretic measures to the sequence of predictions made by the model, resulting in the twin techniques of *agglutination* and *agglomeration*, which elegantly solve the first of the three dilemmas listed above. It is outside the scope of the current discussion to offer solutions to the remaining two; this has been done elsewhere [18].

A. Agglutination

The information provided to a predictive model by a symbol in the data may be considered to be a measure of the degree of independence between this symbol and the one which immediately precedes it, with respect to the model, with low values of information signifying a high degree of correlation between the two symbols.

The *agglutination* process works by finding the most correlated symbol pair in the data, forming a new, higher-level Sub-Object symbol from that pair, UpWriting the data with respect to this new symbol, and iterating until some stopping criterion is reached.[2]

It is possible to view the Sub-Object structure discovered in the data by the UpWrite Predictor at all levels of representation simultaneously in the form of a dendrogram. In figure 8 we show a dendrogram over the first sentence of the Sherlock corpus relative to a character-level UpWrite Predictor inferred from the entire corpus [19]. The hierarchical structure exhibited in this illustration forms a reasonable decomposition of the sentence into phrases and words, with

[1]The lowest-level module in the chain lacks both an alphabet and an UpWriter due to the fact that, at the lowest-level, the alphabet is assumed.

[2]The most basic stopping criterion being "continue until the highest-level representation of the data consists of a single symbol".

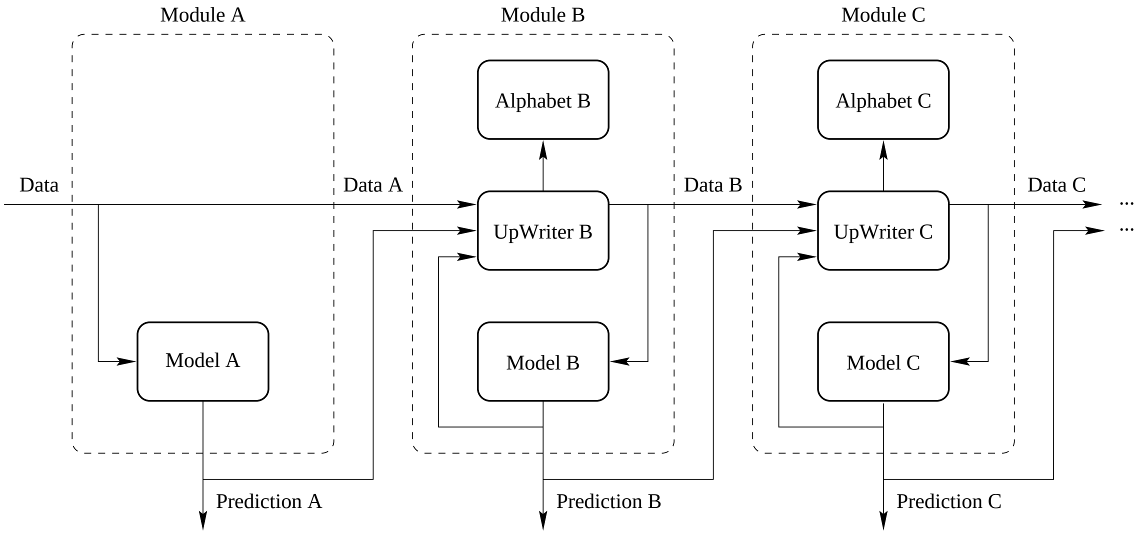

Fig. 7. The structure of the UpWrite Predictor.

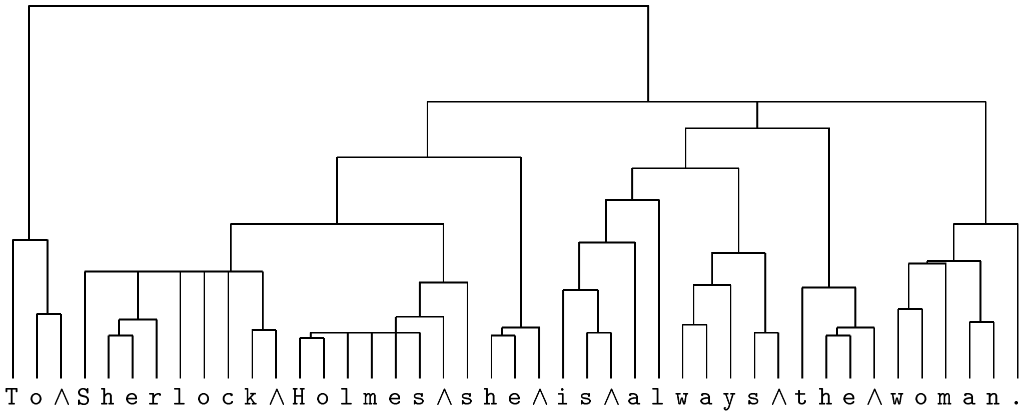

Fig. 8. Dendrogram illustrating a hierarchy of Sub-Objects discovered by the agglutination process, at various levels of representation of the data, relative to a character-level UpWrite Predictor inferred from the Sherlock corpus.

a few obvious mistakes that should be correctable via the feedback mechanism inherent in the model.

B. Agglomeration

Two symbols may very well belong to the same class if the predictive model cannot distinguish between them on the basis of the predictions it makes about which symbols are likely to follow them. Information theoretic measures may be used to calculate the similarity between the predictions made by the model, expressed as probability distributions over the alphabet.[3]

The *agglomeration* process works by finding the two symbols which are the most difficult to tell apart according to this criterion, forming a new Quotient-Object symbol from the pair, UpWriting the data with respect to this new symbol, and iterating.

As with agglutination, it is possible to view, via a dendrogram, the Quotient-Objects found by the agglomeration process at all levels of abstraction simultaneously. Figures 9 and 10 illustrate some of the hierarchical Quotient-Objects found by the algorithm, and it is evident that these Quotient-Objects are sufficiently fine-grained to be quasi-semantic in nature.

[3] We neglect to specify candidate similarity measures here, as the measure chosen is irrelevant as far as the agglomeration method is concerned.

VII. Experimental Results

We shall now evaluate the performance of the UpWrite Predictor on both artificial and natural language data. In both cases, the UpWrite Predictor used will be based upon a 1^{st}–order, character-level Markov model, with agglutination and agglomeration being used to uncover higher-level structure. Performance of the model will be evaluated by measuring the average surprise it experiences upon observing novel data.

A. Artificial Language Data

Consider the phrase-structure grammar shown in figure 11, developed by Gerry Wolff in order to evaluate the performance of his SNPR algorithm [20]. This grammar was used to generate a training corpus 125342 bytes in

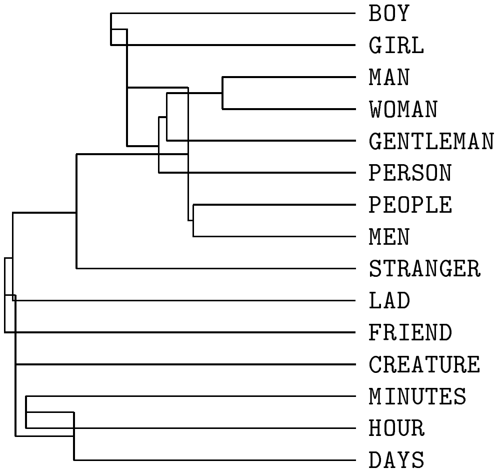

Fig. 9. Dendrogram illustrating a hierarchy of Quotient-Objects discovered by the agglutination process, at various levels of representation of the data, relative to a word-level UpWrite Predictor inferred from the Sherlock corpus.

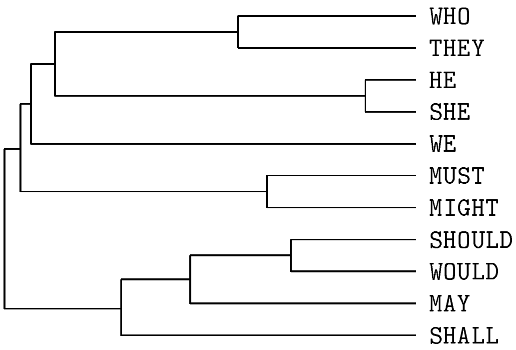

Fig. 10. Dendrogram illustrating a second hierarchy of Quotient-Objects discovered by the agglutination process, at various levels of representation of the data, relative to a word-level UpWrite Predictor inferred from the Sherlock corpus.

length and a testing corpus 12458 bytes in length.

#	↦	(1)(2)(3)|(4)(5)(6)
1	↦	DAVID|JOHN
2	↦	LOVES|HATED
3	↦	MARY|SUSAN
4	↦	WE|YOU
5	↦	WALK|RUN
6	↦	FAST|SLOWLY

Sample: JOHNLOVESMARYDAVIDHATEDMARYYOURUN ...

Fig. 11. The phrase-structure grammar used to generate Text 5 and Text 6.

An UpWrite Predictor was incrementally inferred from the training corpus, and its performance evaluated on the testing corpus whenever a new, higher-level symbol was discovered. The overall performance of the UpWrite Predictor on this data is shown in the plot of figure 12, and it can be seen that a fairly steady improvement occurs up until the thirty-fourth iteration of the algorithm, at which point it levels off.

Some of the structure found by the UpWrite Predictor is listed in table I, together with the iteration in which it was found. It can be seen that the UpWrite Predictor found ten of the twelve symbol sequences implicit in the data by iteration 36, and found four correct symbol classes: [HATED|LOVES], [JOHN|DAVID], [RUN|WALK] and [YOU|WE]. The remaining two symbol classes weren't discovered due to the fact that phrases were formed from the symbol sequences which are members of these classes in iterations 40, 41, 47 and 49. Note that the tested version of the UpWrite Predictor did not feature a mechanism to correct bad generalisations such as these.

4	⟨JOHN⟩
8	⟨HATED⟩
13	⟨DAVID⟩
16	⟨FAST⟩
19	⟨MARY⟩
22	⟨RUN⟩
31	⟨YOU⟩
32	⟨SUSAN⟩
34	⟨LOVES⟩
36	⟨SLOWLY⟩
37	[HATED|LOVES]
38	[JOHN|DAVID]
40	⟨[JOHN|DAVID][HATED|LOVES]SUSAN⟩
41	⟨[JOHN|DAVID][HATED|LOVES]MARY⟩
42	⟨WALK⟩
43	⟨WE⟩
44	[RUN|WALK]
45	[YOU|WE]
47	⟨[YOU|WE][RUN|WALK]FAST⟩
49	⟨[YOU|WE][RUN|WALK]SLOWLY⟩

TABLE I

A LIST OF THE STRUCTURE FOUND BY THE UPWRITE PREDICTOR ON DATA GENERATED BY THE PHRASE-STRUCTURE GRAMMAR OF FIGURE 11, SHOWN IN THE ORDER OF ACQUISITION.

The processes of agglutination and agglomeration form a new symbol from exactly two lower-level symbols. Words such as DAVID are uncovered by the algorithm incrementally, with symbol-pairs found at one level of abstraction serving as "scaffolding" for the extraction of longer symbol sequences.

B. Performance on Natural Language Text

Unfortunately we have not had the opportunity to infer an UpWrite Predictor from a large corpus of natural language text—the inference process is far too expensive in terms of both memory requirements and processing requirements to enable a thorough examination of its performance on natural language. However, we have performed experiments with small natural language corpora, and we present the results of applying the UpWrite technique to natural language text in this section.

A testing corpus was created from the first 99983 bytes of

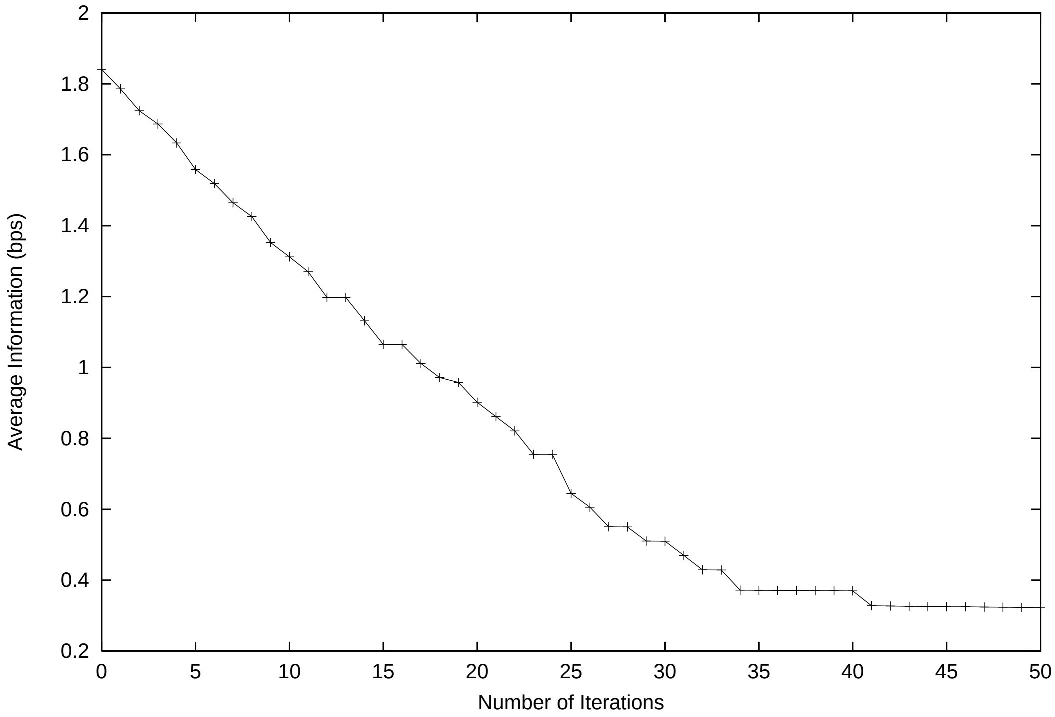

Fig. 12. The performance of the UpWrite Predictor at the end of each iteration as evaluated on data generated by the phrase-structure grammar of figure 11.

the Sherlock corpus, and a training corpus was created from the 1000034 bytes which followed. We plot the performance of the UpWrite Predictor inferred from the training corpus and evaluated on the testing corpus in figure 13, together with the performance of 2^{nd}– and 3^{rd}–order Markov models inferred from and evaluated on the same data.

After 1205 iterations, the performance of the UpWrite Predictor was 27% better than its performance prior to the first iteration, at which stage it was equivalent to a 1^{st}–order Markov model. At iteration 415 of the algorithm, the performance of the UpWrite Predictor exceeded that of the 2^{nd}–order Markov model, and by iteration 1205, the performance of the UpWrite Predictor was a mere 0.8% worse than the performance of the 3^{rd}–order Markov model.

By this stage of the process, the UpWrite Predictor had discovered symbol sequences such as ⟨`Sherlock∧Holmes`⟩, ⟨`Watson`⟩, ⟨`,"∧said∧Holmes.`⟩ and ⟨`and∧the`⟩, as well as symbol sequences corresponding to frequently occurring English words, and a small number of symbol classes.

The fact that a 1^{st}–order Markov model which makes predictions about an UpWritten version of natural language text almost reaches the performance of a standard 3^{rd}–order Markov model suggests that the UpWrite is beneficial, and that the structure found by the UpWrite Predictor is useful in practice.

VIII. Summary and Conclusion

In this paper we have introduced the UpWrite, a powerful framework for the hierarchical extraction of syntax from data. We have given a brief history of the UpWrite, illustrated it via a trivial image classification problem, shown how it can be used in the context of stochastic language modelling, and given experimental results which suggest that it is a powerful modelling technique.

References

[1] David Marr, *Vision*, W.H. Freeman and Company, 1982.
[2] C.J.S. deSilva, M.D. Alder, and Y. Attikiouzel, "Automating knowledge engineering," Tech. Rep. TR90-03, The Centre for Intelligent Information Processing Systems, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1990.
[3] Michael Alder, Christopher deSilva, and Yianni Attikiouzel, "Automatic knowledge acquisition," Tech. Rep. TR90-14, The Centre for Intelligent Information Processing Systems, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1990.
[4] Michael Alder, Christopher deSilva, and Yianni Attikiouzel, "On the automatic generation of higher levels of description," Tech. Rep. TR90-15, The Centre for Intelligent Information Processing Systems, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1990.
[5] Robert A. McLaughlin and Michael D. Alder, "Recognising aircraft: Automatic extraction of structure by layers of quadratic neural nets," in *Proceedings of the IEEE International Conference on Neural Networks*, June 1994, vol. 7, pp. 4288–4293.
[6] Robert A. McLaughlin and Michael D. Alder, "Recognition of infra red images of aircraft rotated in three dimensions," in *Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems*, November 1995, pp. 82–87.
[7] Robert A. McLaughlin, *Intelligent algorithms for finding curves and surfaces in real world data*, Ph.D. thesis, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1999.
[8] Sok Gek Lim, *Visual Object Shape Recognition Using Hierarchical Syntax Extraction*, Ph.D. thesis, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1997.
[9] Gek Lim and Michael D. Alder, "Calvin and Hobbes," Tech.

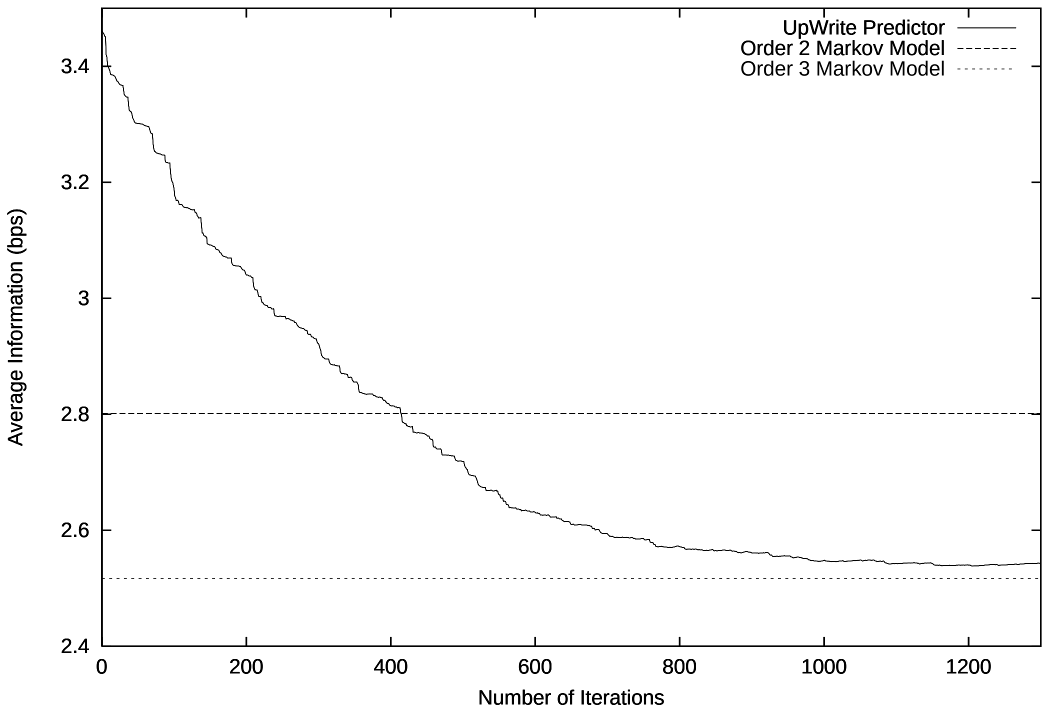

Fig. 13. The performance of the UpWrite Predictor at the end of each iteration as evaluated on natural language text.

Rep. TR95-02, The Centre for Intelligent Information Processing Systems, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1995.

[10] Phillip Dunstan, Gek Lim, and Michael D. Alder, "Real-time head detection," in *Proceedings of the IEEE International Conference on Neural Networks*, November 1995, vol. 5, pp. 2217–2221.

[11] Paul S. Williams, *The automatic hierarchical decomposition of images into sub-images for use in image recognition and classification*, Ph.D. thesis, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1999.

[12] Patrick Chisan Hew, *Pixels to Strokes to Digits*, Ph.D. thesis, Department of Mathematics, The University of Western Australia, Australia 6907, 1999.

[13] Michael D. Alder, Gek Lim, and Christopher J.S. deSilva, "The syntax of images," Tech. Rep. TR95-01, The Centre for Intelligent Information Processing Systems, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1995.

[14] Robert A. McLaughlin and Michael D. Alder, "The Hough Transform versus the UpWrite," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 396–400, April 1998.

[15] Robert A. McLaughlin and Michael D. Alder, "The Hough Transform and the UpWrite: a comparison," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, vol. 1, pp. 146–151.

[16] Robert A. McLaughlin and Michael D. Alder, "Syntactic pattern recognition of simple shapes," in *Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems*, 1993, pp. 5–9.

[17] Robert A. McLaughlin and Michael D. Alder, "Recognising cubes in images," in *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, E. S. Gelsema and L. N. Kanal, Eds. June 1994, number 16, pp. 45–58, Elsevier Science B.V.

[18] Jason L. Hutchens, *The UpWrite Predictor: A General Grammatical Inference Engine for Symbolic Time Series with Applications in Natural Language Acquisition and Data Compression*, Ph.D. thesis, Department of Electrical & Electronic Engineering, The University of Western Australia, Australia 6907, 1999.

[19] Sir Arthur Conan Doyle, "The Sherlock/SHERLOCK corpora," These corpora were formed by Jason Hutchens from a collection of texts downloaded from the Oxford Text Archive, and are available from:
`http://ciips.ee.uwa.edu.au/~hutch/sherlock/`, 1995.

[20] J. Gerard Wolff, "Language acquisition, data compression and generalization," *Language & Communication*, vol. 2, no. 1, pp. 57–89, 1982.

[21] Michael D. Alder and Christopher deSilva, "Topological stochastic grammars," in *IEEE International Symposium on Information Theory*, July 1994.

[22] Michael D. Alder, "Inference of syntax for point sets," in *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, E. S. Gelsema and L. N. Kanal, Eds. June 1994, number 16 in Machine Intelligence and Pattern Recognition, pp. 45–58, Elsevier Science B.V.

[23] King-Sun Fu and Taylor L. Booth, "Grammatical inference: Introduction and survey - part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 3, May 1986.

[24] King-Sun Fu and Taylor L. Booth, "Grammatical inference: Introduction and survey - part II," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 3, May 1986.

[25] Claude E. Shannon and Warren Weaver, *The Mathematical theory of Communication*, University of Illinois Press, 1949.