Banish Prototype Description and Roadmap

Reference: 0001090, March 2014

Authors: R. J. A. Tough and T. M. Cooper,

Issue Record

The content of this document is defined by the issue number and, where appropriate, the revision identifier detailed below.

Issue / Revision	Incorporated by	Date	Comments
1.0	Tom Cooper	31/03/14	End of Phase A

Authorisation

Prepared by: T.M. Cooper, R.J.A.Tough, Igence Radar Ltd

Approved by: Paul Shepherd, Igence Radar Ltd

Contents

1. Executive Summary

1.1 Contractual

1.1.1 The work described in this report was carried out for MOD under contract DSTLX1000090055. This report is offered in partial fulfilment of milestone 1: “Phase A” – Software prototype and technology roadmaps.

1.2 Technical context

1.2.1 The potential benefits of the application of Bayesian techniques to intelligence are well understood and have been articulated by the Head of Future Analytic Methods at Defence Intelligence (DI). A critical challenge for the adoption of these techniques is to develop a solution that can be inserted into analysts workflow and provide a compelling user experience with clear benefits to the analyst.

1.2.2 In order to demonstrate how Bayesian techniques can be included within standard DI practice it is critical for the project team to interact with the DI analyst community; both to understand their work practices and to ensure that their requirements are embodied in the user interface design and user experience.

1.3 Scientific and technical progress

1.3.1 A prototype software tool has been produced to demonstrate how Bayesian techniques can be easily included within standard DI practice. The prototype tool is a C++ application that can run on the Microsoft Windows OS (Windows 7, Vista and XP). The work has focused on the User Interface and User Experience by capturing the practical requirements of the analysts and embodying them in the software interface.

1.3.2 An initial user interface design was constructed informed by the three example problems (Two Jars, Diagnostic Cards and Red's CW), the initial requirements capture provided in the powerpoint briefing pack and the outcome of the kick-off meeting. Once this was in place, feed-back was sought from Dstl and DI analysts and the results informed a second cycle of development.

1.3.3 In addition to the prototype software, a set of technology route maps has been produced. These include three route maps listed in the requirements document and additional analysis of how the software might be modified to cater for non-standard analysis inputs.

1.3.4 The aim of the tree original route maps is to assess the utility of three techniques to extend the usefulness of the prototype software:

- The potential of crowd sourcing the *a priori* probabilities for the system;
- The potential use of graph similarity measures to construct dynamic teams;
- The potential use of taking extracted triples from unstructured documents as alerting data for appending to the evidence in a network.

1.3.5 In each case, detailed technical requirements were captured from Dstl and an assessment of the utility of each of the potential technologies was made identifying the technical developments required to incorporate these techniques in the existing software tool (or as stand-alone tool) if appropriate.

2. Introduction

2.1 Technical context

2.1.1 The potential benefits of the application of Bayesian techniques to intelligence are well understood and have been articulated by the Head of Future Analytic Methods at Defence Intelligence (DI). A critical challenge for the adoption of these techniques is to develop a solution that can be inserted into analysts workflow and provide a compelling user experience with clear benefits to the analyst.

2.1.2 In order to demonstrate how Bayesian techniques can be included within standard DI practice it is critical for the project team to interact with the DI analyst community; both to understand their work practices and to ensure that their requirements are embodied in the user interface design and user experience.

2.2 Scientific and technical progress

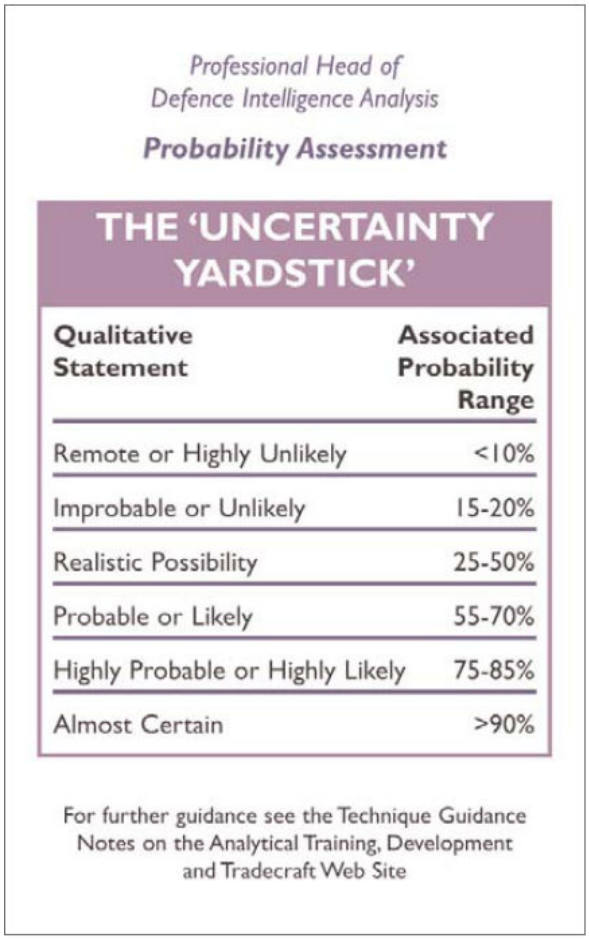
2.2.1 A prototype software tool has been produced to demonstrate how Bayesian techniques can be easily included within standard DI practice. The prototype tool is a C++ application that can run on the Microsoft Windows OS (Windows 7, Vista and XP). The work has focused on the User Interface and User Experience by capturing the practical requirements of the analysts and embodying them in the software interface.

2.2.2 An initial user interface design was constructed informed by the three example problems (Two Jars, Diagnostic Cards and Red's CW), the initial requirements capture provided in the powerpoint briefing pack and the outcome of the kick-off meeting. Once this was in place, feed-back was sought from Dstl and DI analysts and the results informed a second cycle of development.

2.2.3 The initial design process defined the formats for the import and export of Bayes Nets and structured data using open source standards. Consequently, the software allows saving to a JSON file and export to GraphML and bmp. The use of open source libraries was preferred to the production of bespoke software when the library provided the functionality and flexibility required for the application. Consequently the software uses the ‘rapidjson’ JSON parser/generator library and the Open Graph Drawing Framework (OGDF) library for the layout of the Bayesian net diagrams.

2.2.4 The User Interface design provides a project workspace and the capability to access previous judgments from a palette window. When evidence or judgments are added, the software will provides the ability to record supporting information such as the source of the evidence. Metadata is recorded to document changes to the project tagged with user ID, date and time to provide an audit trail for the construction of the Bayes net. The user interface also provides the ability to save and restore projects to support collaborative working.

2.2.5 The key element in the construction of a Bayes net is the population of the probability tables or distribution parameters. The user interface provides the ability to populate these tables using numerical values or through an "uncertainty yardstick" based on that shown below. When the uncertainty yardstick is used for input, the central value of the selected range will be used in the probability table.



Professional Head of
Defence Intelligence Analysis

Probability Assessment

THE 'UNCERTAINTY YARDSTICK'

Qualitative Statement	Associated Probability Range
Remote or Highly Unlikely	<10%
Improbable or Unlikely	15-20%
Realistic Possibility	25-50%
Probable or Likely	55-70%
Highly Probable or Highly Likely	75-85%
Almost Certain	>90%

For further guidance see the Technique Guidance Notes on the Analytical Training, Development and Tradecraft Web Site

Figure 2-1: Uncertainty yardstick

2.2.6 In addition to the prototype software, a set of technology route maps has been produced. These include three route maps listed in the requirements document and additional analysis of how the software might be modified to cater for non-standard analysis inputs.

2.2.7 The aim of the original route maps is to assess the utility of three techniques to extend the usefulness of the prototype software:

- The potential of crowd sourcing the *a priori* probabilities for the system;
- The potential use of graph similarity measures to construct dynamic teams;
- The potential use of taking extracted triples from unstructured documents as alerting data for appending to the evidence in a network.

2.2.8 In each case, detailed technical requirements were captured from Dstl and an assessment of the utility of each of the potential technologies was made identifying the technical developments required to incorporate these techniques in the existing software tool (or as a stand-alone tool) if appropriate.

2.2.9 A literature survey of crowd sourcing methodologies has been produced, and a number of options for how a set of opinions might be combined in this context are described. It is concluded that the design of graph similarity based tools would be premature before there have been any real world models constructed that might benefit from them. Finally, it was decided that there is probably little advantage to be gained from automating triple lookup cued from the Bayes net design compared to a

simple independent system that would allow an analyst to manually search a triple database for term combinations of interest (together with synonyms).

2.2.10 The feasibility of a number of desirable extensions to the software is discussed. These would allow a user or users more flexibility in terms of the data they input to the model and in some instances would provide feedback to the users about the usefulness of data in the form that the users have provided.

3. Implementation description

3.1 Functionality

3.1.1 Detailed instructions for using the prototype software are contained in the user guide. In this section we attempt to convey an intuition for the meaning and utility of the calculations the tool can perform.

3.1.2 The tool allows a directed acyclic graph so be specified by the user. The user names each node in the graph and specifies a set of named values for that node (for example ‘true’ and ‘false’). After this the user should specify ‘conditional’ probability distributions for each node, for each possible combination of values of its parents. The final step in specifying a scenario is to ‘observe’ the values for some nodes. This represents the situation in which the value of some variable is known.

3.1.3 After a scenario has been set up in the software, a few different queries can be conducted. The first is that ‘marginal’ distributions for the nodes can be calculated. These are probability distributions: one for each node that say how likely it is that the node takes each of its possible values.

3.1.4 The second query the software enables is to calculate the entropy (also known as Shannon information) for each node. This non-negative quantity is measured in bits, and describes how much information about that node is unknown. It is a function of the marginal distribution for the node. For a two value node, the entropy is maximised if each value has probability 0.5, in which case the entropy is 1 bit. For a four value node, the entropy is maximised if each value has probability 0.25, in which case the entropy is 2 bits. An eight value node has a maximum entropy of 3 bits etc. A node for which the value is almost certain has a low entropy because intuitively, there isn’t much information about that node that is not known. To put it another way, unless the true value of an almost certainly known node is of vital importance, it is probably not worth investing much effort to confirm it 100%. The entropy of a node of known value (for example an observed node) is zero bits.

3.1.5 The third query the software enables is to calculate the mutual information between nodes. The mutual information is a quantity defined for any pair of nodes, and is also a non-negative quantity measured in bits: intuitively it tells you how much information you would expect to learn about one node from discovering the value of a second node. Mutual information is symmetric in the sense that the mutual information between node A and node B is the same as the mutual information between node B and node A. The mutual information between two nodes cannot be greater than the entropy of either node, and if the mutual information between nodes A and B is equal to the entropy of node B, then discovering the value of node A will let you deduce the value of node B. At the other extreme, the mutual information between two nodes is zero if and only if the nodes are statistically independent: that is if discovering the value of node A would tell you no information about node B.

3.1.6 Displaying the mutual information between all nodes and a node of interest will show how much information would be expected to be discovered about that node by collect operations on other nodes, i) in an absolute sense, ii) in comparison to the amount of information that remains to be discovered about the node of interest, and iii) enabling comparison between the value of various possible single node collects. To discover how much information would be expected to be gained about a node of interest from a set of collects, a new node representing the collect set should be added to the network. To discover how much information would be expected to be gained about a set of nodes of interest, a new node representing the concatenation of the nodes of interest should be added to the network.

3.1.7 Displaying the mutual information between all nodes and a collect node will show how much information would be expected to be discovered about each node from a collect operation.

3.1.8 An example is shown in Figure 1, which shows a Bayesian net representation of a question about cards. The question is as follows. There is a set of four cards, each of which has a letter written on one side and a whole number written on the other. The cards are laid out on a table showing the text: 'A', 'B', '3' and '4' respectively. We wish to find out the truth of the statement that all the cards with a vowel on one side have an even number on the side. Which cards should we turn over to discover the truth of that statement?

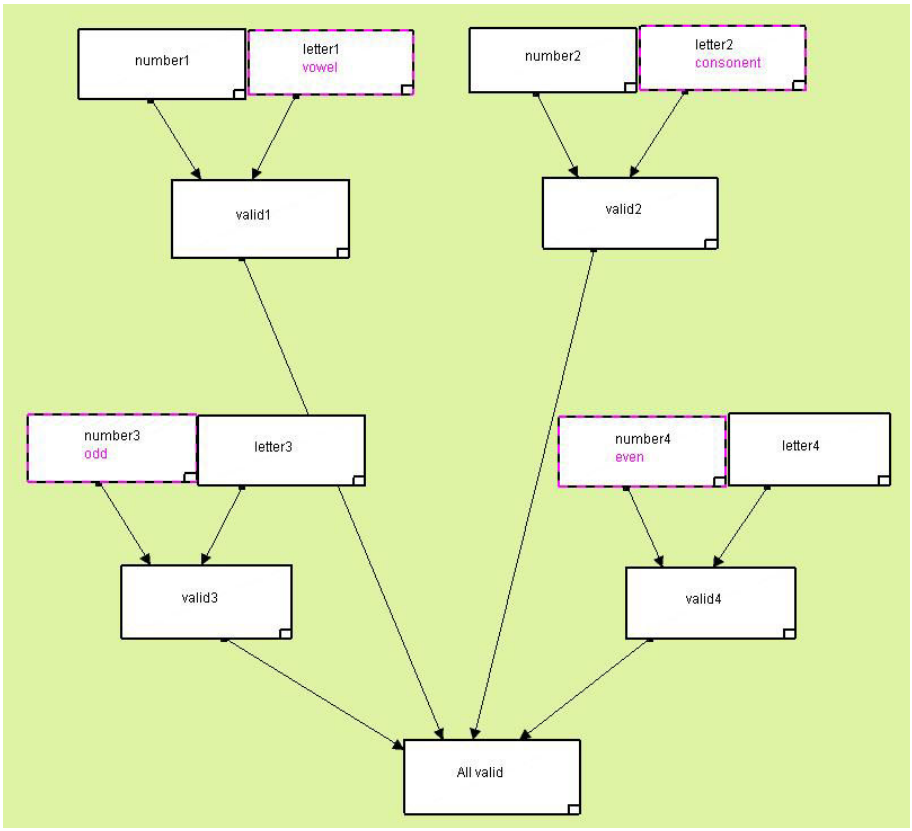


Figure 1 A Bayes net for the cards question

3.1.9 In the figure, the four cards are each represented by a pair of nodes depicting the number and letter on the card together with a node representing whether the card satisfies the condition of interest. There is an additional node representing whether all the cards satisfy the condition of interest. The prior distribution for whether each card's letter is a vowel or a consonant is set at (0.5, 0.5), as is the distribution for whether each card's number is odd or even, although the same qualitative results would be obtained with any intermediate probability. The validity nodes have a probability of one of being true if the corresponding number is even or if the corresponding letter is a consonant, and a probability zero of being true otherwise. The 'All valid' node has a probability of one of being true if the four validity nodes are all true, and a probability zero of being true otherwise.

3.1.10 The top left card is observed as having a vowel, the top right card is observed as being a consonant, the bottom left card is observed as having an odd number, and the bottom right card is observed as having an even number. Figure 2 shows the resulting mutual informations between the 'All valid' node and each of the other nodes.

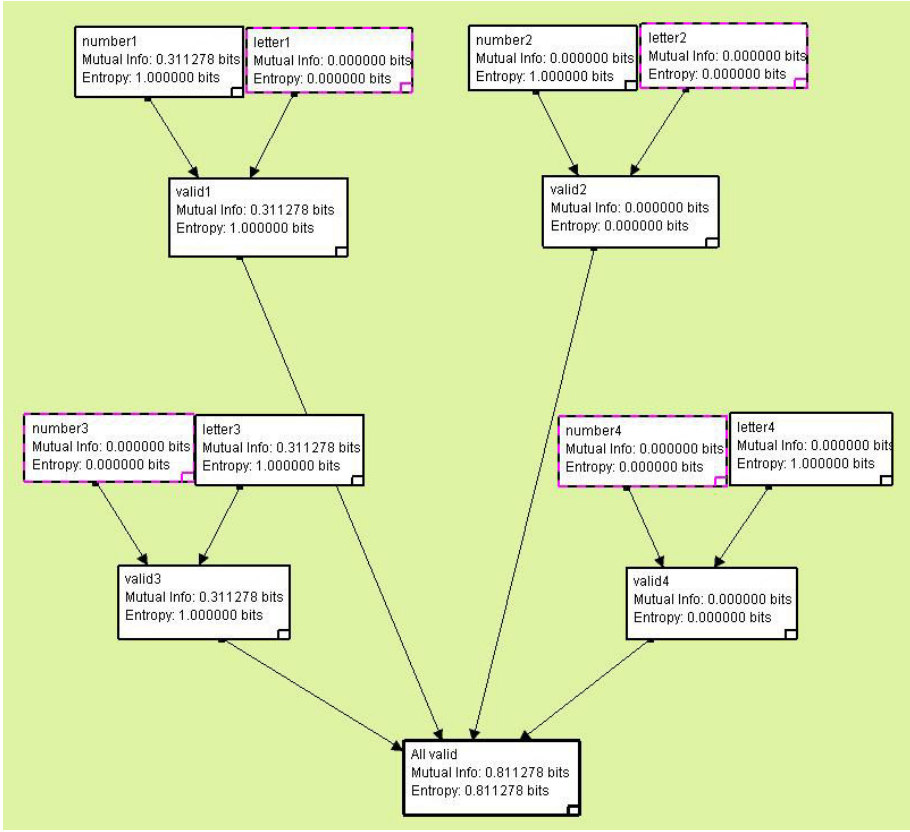


Figure 2 Mutual informations for the card question

3.1.11 Figure 2 shows that there is no entropy (unknown information) in the (purple) nodes representing the observed side of each card and that there is exactly one bit of entropy in the node representing the unobserved side of each card. Of these unobserved sides, the top right and bottom right will give you no information about whether all the cards are valid, and the top left and bottom left will each give you about 0.31 bits of information whether all the cards are valid.

3.1.12 Interestingly, the entropy of the node representing whether all the nodes are valid is about 0.81, which is more than the sum of the entropies of the hidden card sides. This paradox can be interpreted as meaning that there is a degree of synergy

between the missing observations: for the purpose of finding whether all the cards are valid, knowing both relevant hidden sides gives more than twice as much information as knowing either one.

3.1.13 Perhaps a clearer version of this paradox occurs in the problem of determining whether two coin tosses have the same result. Knowing the result of either coin toss gives no information about the question of interest, but knowing the result of both determines the answer absolutely.

3.2 API

3.2.1 The Bayesian net calculations of marginal distributions and mutual informations are performed in a DLL. The interface for the DLL is a C++ header file quoted here

```
#pragma once

#include "BayesNetEngineExports.h"
#include "BayesNetEngineTypes.h"

#include <vector>

class BayesNetImpl;

//The information the BayesNet class stores can be divided into five types,
//with a complex dependency structure (itself a DAG).
//The first (bottom) type is which nodes exist.
//The second type is which arrows exist, this depends on which nodes exist.
//The third type is the number of values each node has, which again depends
//on which nodes exist.
//The fourth type is the conditional probability distributions for each node,
//known here as the node's 'population'.
//This type depends on which arrows exist, and on the number of values each
//node has.
//The fifth type is the observations made (of some nodes), if any.  This type
//depends on the number of values each node has.
//The dependencies are illustrated in this diagram.

// C   O
// | \ |
// A   V
//  \ /
//   N

//Setters for a type fail if the new information contradicts
//information held about a lower type.  Setters for a type typically delete
//information that depends on that type held about that part of the graph.

//Algorithms can only be run on the graph once all levels of information are
//specified (although for nugatory definitions, the empty (ie original)
//specification is correct).

class BayesNet
{
public:
   BAYESNETENGINE_API BayesNet();
   BAYESNETENGINE_API ~BayesNet();
   BAYESNETENGINE_API BayesNet(const BayesNet &other);
   BAYESNETENGINE_API BayesNet &operator=(const BayesNet &other);

////////////////////////////Setters/////////////////////////////////////////
 //

   BAYESNETENGINE_API int addNode(NODENO nodeNo);
   //Fails with 1 if this node number already exists.
   //Returns 0 on success.

   BAYESNETENGINE_API int removeNode(NODENO nodeNo);
   //Fails with 1 if node does not exist.
   //Removes observation and population if they exist,
   //and any arrows to or from the node.
   //Also removes population for any child nodes.
   //Returns 0 on success.

   BAYESNETENGINE_API int specifyNoValues(
      NODENO nodeNo,unsigned int noValues);
   //Fails with 1 if this node number does not exist,
   //Removes any populations and observations that exist for this node,
   //and removes populations from all child nodes.
```

```
   //Returns 0 on success.

   BAYESNETENGINE_API int unspecifyNoValues(NODENO nodeNo);
   //Fails with 1 if node does not exist.
   //Fails with 2 if node's noValues are not specified
   //Removes any populations and observations that exist for this node,
   //and removes populations from all child nodes.
   //Returns 0 on success.

   BAYESNETENGINE_API int addArrow(
      NODENO sourceNodeNo, NODENO destinationNodeNo);
   //Fails with 1 if either of the nodes don't exist.
   //Fails with 2 if this arrow already exists.
   //Fails with 3 if a cycle would be created.
   //Erases destination node population if it exists.
   //Returns 0 on success.

   BAYESNETENGINE_API int removeArrow(
      NODENO sourceNodeNo, NODENO destinationNodeNo);
   //Fails with 1 if either of the nodes don't exist.
   //Fails with 2 if the arrow doesn't exist.
   //Erases destination node population if it exists.
   //Returns 0 on success.

   BAYESNETENGINE_API int populateNode(
      NODENO nodeNo,
      const std::vector<std::vector <double>> &distribution);
   //Fails with 1 if the node doesn't exist.
   //Fails with 2 if the node's noValues doesn't exist.
   //Fails with 3 if any of the node's parents' noValues don't exist
   //Fails with 4 if distribution doesn't meet the following criteria
   //distribution.size() is n1 x n2 x ... x nm
   //where ni is the number of elements for the ith parent node.
   //The parents here are ordered as in the result of getParentNodes().
   //The indexing into distribution is such that the nm index changes
fastest.
   //Each element of 'distribution' should have the same size,
   //which is the number of elements for this node.
   //Each element of each element of 'distribution' should be between 0 and
1.
   //The sum of the elements in each element of 'distribution' should be 1.
   //Creates a new distribution or overwrites old one.
   //Returns 0 on success.

   BAYESNETENGINE_API int depopulateNode(
      NODENO nodeNo);
   //Fails with 1 if node doesn't exist
   //Fails with 2 if node doesn't have a distribution.
   //Erases the node's distribution.
   //Returns 0 on success.

   BAYESNETENGINE_API int observeNode(
      NODENO nodeNo,
      unsigned int value);
   //Fails with 1 if node doesn't exist.
   //Fails with 2 if node doesn't have a noVals.
   //Fails with 3 if value is greater than or equal to the number
   //of elements for this node.
   //Specifies a new observation or overwrites an old one.
   //Returns 0 on success.

   BAYESNETENGINE_API int unobserveNode(
      NODENO nodeNo);
   //Fails with 1 if node doesn't exist.
   //Fails with 2 if node doesn't have a noVals.
   //Fails with 3 if the node has not been observed.
   //Removes an observation of a node.
   //Returns 0 on success.

   BAYESNETENGINE_API int calculateMarginals();
   //Fails with 1 if some node in the net does not have a distribution.
   //'marginals' is populated with a vector<double> for each node in the
   //same order as the result of getNodes.
   //The number elements in this vector<double>
   //is the number of elements for the corresponding node, each element
   //is greater than or equal to 0 and their sum is 1.
   //Use BayesNetEngine::getMarginal() to access the result.
   //Returns 0 on success.

   BAYESNETENGINE_API int calculateMutualInfo(NODENO nodeNo);
   //Fails with 1 if some node in the net does not have a distribution.
   //Use BayesNetEngine::getMutualInfo() to access the result.

  ////////////////////////////Getters/////////////////////////////////////////
   //
```

```
   BAYESNETENGINE_API std::vector<NODENO> getNodes()const;
   BAYESNETENGINE_API bool noValuesSpecifiedP(NODENO nodeNo)const;
   BAYESNETENGINE_API unsigned int getNoValues(NODENO nodeNo)const;
   BAYESNETENGINE_API std::vector<NODENO> getChildNodes(
      NODENO nodeNo)const;
   BAYESNETENGINE_API std::vector<NODENO> getParentNodes(
      NODENO nodeNo)const;
   BAYESNETENGINE_API bool distributionSpecifiedP(NODENO nodeNo)const;
   BAYESNETENGINE_API const std::vector<std::vector<double>> *getPopulation(
      NODENO nodeNo)const;
   BAYESNETENGINE_API bool nodeObservedP(NODENO nodeNo)const;
   BAYESNETENGINE_API unsigned int getObservedValue(NODENO nodeNo)const;
   BAYESNETENGINE_API bool marginalsExistP()const;
   BAYESNETENGINE_API const std::vector<double> *getMarginal(
      NODENO nodeNo)const;
   BAYESNETENGINE_API  double  getMutualInfo(NODENO  nodeNo)const;//Result  in
bits
   BAYESNETENGINE_API NODENO getUnusedNodeNo()const;

   BAYESNETENGINE_API int getNetStatus()const;
   //Returns 1 if some noValues is not defined, otherwise
   //returns 2 if some population is not defined, otherwise
   //returns 3 if the marginals don't exist
   //(but are ready to be calculated) otherwise
   //returns 4.

private:

   BayesNetImpl *impl;
};

BAYESNETENGINE_API double entropyFromProbDist(
   const std::vector<double> &probDist);
//The elements of probDist should be in [0,1] and sum to 1.
//Returns the entropy in bits of the probability distribution.
```

3.2.2 The engine has no knowledge of node or value names or of the locations of nodes in the displayed diagrams. Instead each node is specified by a unique number. All the operations necessary to define and query the topology of the network and the corresponding conditional distributions are provided together with operations to calculate and query the resulting marginal distributions and mutual informations.

3.3 Internal structures and calculations

3.3.1 Internally the Bayesian net is stored as a set (`stl::set`) of node numbers, together with associative arrays (`stl::map`) storing the network topology (targets indexed by arrow sources and sources indexed by and arrow targets), the conditional distributions (indexed by node number) and the observations (also indexed by node number).

3.3.2 Currently the main calculations carried out by the prototype software are the marginal distributions for each node and the mutual information between each node and a selected node.

3.3.3 The marginal distributions are computed simply by iterating exhaustively through all possible combinations of values for the complete set of nodes incrementing an appropriate accumulator for each combination and node. Finally the accumulated values are normalised by dividing by the sum for that node so that the marginal values for each node sum to one.

3.3.4 The computational complexity of this algorithm is exponential in the number of nodes, so it would be impractical for large networks (above about a dozen nodes). The reason this implementation was chosen over a message passing algorithm is that the exhaustive algorithm gives an exact answer rather than sometimes giving an approximation, that it is comparatively simple to code, and that it is more likely to be compatible with future augmentations to the software functionality. This implementation choice might have to be revisited for later versions of the software.

3.3.5 The first stage of calculating the mutual information between two nodes is to calculate the joint distribution of the two nodes. This is calculated in the same way as the single node marginals: all possible combinations of the complete set of nodes are iterated through and an accumulator is incremented for each combination. After the unnormalised joint probability distribution has been calculated, the mutual information between the two nodes is computed using the formula

$$\text{mutual information} = \sum_x \sum_y \frac{q_{xy}}{\sum_x \sum_y q_{xy}} \left[\ln(q_{xy}) - \ln\left(\sum_x q_{xy} \right) - \ln\left(\sum_y q_{xy} \right) + \ln\left(\sum_x \sum_y q_{xy} \right) \right]$$

here q_{xy} is the unnormalised joint probability distribution of the two random variables *x* and *y*.

4. Soliciting expert opinion

4.1 Introduction

4.1.1 The past three decades have seen enormous progress in the use of Bayesian methods, which employ probabilistic formalism and computational tools, in the analysis of and drawing inferences from incomplete and uncertain knowledge derived from the necessarily limited observations available in practice. These developments proceeded through a period of epistemological controversy, in which the validity of the application of probabilistic methods to such problems was hotly contested, particularly by members of the so-called 'frequentist' school of thought; the appropriation and further development of computational and approximate analytical methods from statistical, computational and physical science that overcame the seeming intractable problems that arose in their initial implementation; and their successful application to questions posed by ever larger and more widely sourced bodies of data. The potential of these methods in the analysis of intelligence problems has been recognised for many years; preliminary discussions and demonstrations of the Bayesian methodology in this context pre-dated their wide-spread adoption in the scientific community at large. An account of some of this historical and methodological background is given in the report [1]; more detailed discussions of the principles underlying Bayesian analysis can be found in the texts [2] and [3]. Some early intelligence applications were subseqeuntly de-classified and published in [4] and [5]

4.1.2 One of the strengths of the Bayesian methodology is its systematic and explicit incorporation of the available knowledge in terms of probabilistic models (see Chapter 3 in ref [3]). Conversely any serious shortcomings in these models will compromise the outputs of the Bayesian analysis quite significantly and can vitiate the conclusions drawn from them. In many applications, such as the problem of medical diagnosis described in [1], the prior and conditional probabilities accommodated in a Bayes net are constructed on the basis of extensive data analysis and, in order to take account of the more subjective aspects of expert judgement, from the opinions of established practitioners with considerable experience in addressing the problem under consideration. In this section we will consider several ways in which this expert opinion is solicited. The classic approach, which was used in early intelligence applications of Bayes methods, is the so-called Delphi method, in which the canvassing of the views of a set of experts is carried out through the use of anonymous questionnaires and an iterative refinement through moderation and review. More recently, since the advent of the internet, other methods of collaborative working have emerged that may be useful in the construction of reliable statistical models; we will consider two of these options, the expert Wiki and crowd-sourcing, and compare them with the Delphi approach.

4.2 The Delphi Method

4.2.1 At the time of the studies described in [4] and [5] the effective sampling of the opinions, of experts and the general population alike, had already achieved a reasonable degree of sophistication, and was used extensively for both commercial and political purposes. One particularly effective strategy, the so-called Delphi process, had been developed by the RAND corporation for the soliciting, refining and combining of expert opinions and had been applied, for example, to the discernment of directions of current and future scientific and technological progress. Its methodology and format were specifically designed to transcend inter-disciplinary, cultural and personal barriers and to facilitate creative thought and seemingly serendipitous cross-fertilizations. The principal features of this technique were:

(i) Its framing of problems in quantitative terms

(ii) The identification and assembly of a panel of experts, whose anonymity is protected

(iii) The interrogation of these experts through interview and questionnaire

(iii) The controlled iteration of its outputs, expressed in numerical form and with the anonymity of their sources assured, is carried out under the supervision of the facilitator until a consensus emerges

4.2.2 Thus the Delphi process was able to elicit, refine and summarise the opinions of a large number of experts without encountering problems associated with group dynamics and institutional and personal bias. Initial applications of the methodology were in the main to the forecasting of technological and scientific progress and their sociological, political and strategic impact; [6] provides an early, and rather harrowing, example of studies of this sort. Subsequent applications to business and commercial forecasting and to policy formulation were generally considered to be successful; [7] provides a very full account of these and other developments. The Delphi approach was also adopted, with suitable modifications, to assemble the probabilistic building blocks required for the Bayesian analysis described in [5].

4.2.3 The incorporation of the input of a number of experts into this knowledge base had several advantages, among them being its bringing to bear a greater range of competence than that of any one individual, the resulting capture of a more exhaustive set of causes and effects, and a mitigation of institutional and personal bias. Preserving the anonymity of the experts during this initial information gathering phase and the initial processing of their inputs, in their absence and by the facilitator, also helped eliminate any effects borne of personal and professional animus. Once this preliminary assessment and documentation, typically in a concise and, if possible, quantitative format, of the factors involved had been prepared it was assessed, again under conditions of anonymity, by the experts; if necessary this reviewing and refinement could be repeated iteratively until a satisfactory knowledge base was achieved.

4.2.4 The original implementations of the Delphi method were rather labour intensive, with hard copy documents to be compiled, printed and reviewed, and each stage having a significant turnaround time. Modern communication and data processing technologies have rendered much of this redundant and the Delphi method is now a more streamlined and efficient process. The rounds of review can be replaced by a process of continuous interaction, with real time updating of the emerging conclusions. In some contexts the open access provided by the internet would furnish the investigation with a larger pool of opinions, which can be an advantage; this is not likely to be the case in intelligence applications where considerations of security may be paramount.

4.3 Expert blogs and Wikis

4.3.1 Perhaps the most distinctive feature of the Delphi method is its use of a panel of experts who are appointed and anonymous; neither of these characteristics is typical of a member of an on-line collaboration hosted by a blog and its associated expert Wiki. Such collaborations tend to address demanding problems whose solutions has significant kudos attached to them, which, in addition to interest in the problem in hand, provides some of the motivation for the participants' involvement.

4.3.2 A particularly striking example of this approach is provided by the polymath project [8] which addresses mathematical problems in a massively collaborative fashion. Anyone is welcome to contribute to these joint efforts, but all are expected to abide by rules that attempt to foster the serendipitous cross fertilization of as yet imperfectly formed or expressed ideas

4.3.3 In outline these rules of the polymath project stipulate that

(1) Everyone (regardless of mathematical level) is welcome to participate as even very simple or “obvious” comments, or comments that help clarify a previous observation, can be valuable.

(2) Participants are not to treat the project as a race between individuals; the purpose of the polymath project is to solve problems collaboratively rather than individually, by proceeding via a multitude of small observations and steps shared between all participants

(3) Once the number of comments has become too large to easily digest at once, participants should work on a wiki page to summarise the progress made so far, to help others get up to speed on the status of the project.

(4) Any non-research discussions regarding the project (e.g. administrative suggestions, or commentary on the current progress) should be made at a separate discussion thread, to ensure that the collaborative effort is not diluted

(5) Participants’ comments should be polite and constructive and be as easy to understand as possible, and bear in mind that the mathematical level and background of other participants may vary widely.

4.3.4 Some quite spectacular progress has been achieved through collaborations performed subject to these guidelines. While this may in part be due the exceptional caliber of some of the participants, who are particularly interested in fostering this novel form of collaborative working and whose non-anonymous presence doubtless attracts many of the other participants to the venture, the more relaxed and freewheeling approach adopted within this framework by a self-selected team of experts with a real personal motivation to solve the problem in hand, does appear to be a successful strategy for the solution of non-routine, difficult, and frequently open-ended, intellectual problems. These, unlike the long term forecasting frequently generated by the Delphi method, do have recognizably correct solutions that can be identified by a consensus of the participants. While the polymath project is completely open and accessible in a way that would not be suitable to the solution of intelligence problems a similarly informal framework promoting serendipitous and intuitive interactions within a secure but nonetheless reasonably large and diverse population of analysts could be very effective. In this context it is interesting to note that, while expert anonymity is a defining feature of the Delphi approach, the early studies [4], [5] did comment that useful insights emerged when the participating analysts were ‘unmasked’ and that these were of considerable interest to the participants themselves.

4.4 Crowdsourcing

4.4.1 The ethos of the essentially academic polymath project has its counterpart in the open source development of software, whose outputs are available to and of great benefit to a much wider community. A globally distributed assembly of expertise is now being accessed rather differently, and with more commercial intent, through the mechanism of crowdsourcing. While this is enabled by the technology of the internet, it is inspired by the recognition that a 'crowd' of many dispersed individuals excels at singular, sometimes highly complex problems for which more traditional problem-solving strategies fail. Surowiecki [9] examines several cases of crowd wisdom at work, where the very success of a solution is dependent on its emergence from a large body of solvers. Based on these empirical investigations – from estimating the weight of an ox, to the Columbia shuttle disaster, to gaming sports betting spreads– he finds that 'under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them'. This 'wisdom of crowds' is derived not from averaging solutions, but from aggregating them.

4.4.2 Building on this insight crowdsourcing has been developed as a problem solving strategy in which a company or institution takes a function once performed by employees and outsources it to an undefined (and generally large) network of people through the medium of a public announcement or open call. The response to this can take the form of peer-production, when the job is performed by an ad hoc collaboration; it can also be undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers. Several tasks are well suited to this form of out-sourcing, particularly those that the human operative can perform better or more cheaply than a computer, robot or in-house staff. Creative design tasks have been effectively crowdsourced, as have the provision of commercial imagery, industrial R and D tasks, geological prospecting and large scale searches of satellite and other mapping systems for distinctive features that the human eye/brain combination can recognize. Within a commercial context, however, a crowdsourcing operation cannot be regarded as complete unless it successfully yields an output that is monetized by the organization that initiates the activity [10].

4.4.3 A recognized shortcoming of this crowdsourcing approach to problem solving and other tasks is its vulnerability to subversion; its ability to address and access an enormous potential workforce also exposes it to a great deal of potential mischief. In the intelligence context this weakness must, it would seem, be a very serious and almost unavoidable disqualification from fitness for purpose. Nonetheless an accessing of this wisdom of crowds must be a potentially invaluable intelligence tool. In the Delphi method the anonymity of the experts was preserved to facilitate their interaction; in this case it is the identity of the instigating organization or 'employer' that must be disguised. In this context the internet does not so much provide the work force but rather the enormous body of opinions the crowd unwittingly expresses through its members' transactions, communications and, increasingly, their social networking activities. The characterization and analysis of this massive traffic of communications is currently a topic of both active research and controversy. So, for example, there is a large literature on using text from various online sources for prediction of economic indexes and stock market trends (see [11] for instance). In general the methodology of these studies is to obtain natural language text from the Web, such as news stories, message board data, Twitter feeds, etc., and to use language specific features to train a classification algorithm to predict the future direction or value of the index/market. The adaptation of this approach to intelligence applications, while it is doubtless under development, would have us address technical and ethical issues that are beyond the purview of a preliminary report such as this.

4.5 Conclusions

4.5.1 The success of any Bayesian analysis is crucially dependent on the fidelity of the probabilistic models it employs. Here we have considered how the information underpinning such models might be assembled from the inputs of a body of experts.

4.5.2 A well established solution to this problem is the Delphi method, in which a panel of experts is selected and interrogated through questionnaire and interviews. Their opinions are analysed and summarized by a facilitator and re-submitted to the panel, under conditions of strict anonymity, to the panel for reconsideration. This process is repeated iteratively until a consensus is achieved. A self-selected and identifiable panel of experts can be assembled and accessed effectively through a blog/Wiki format, though considerations of security may become more problematic for intelligence applications of this more informal approach.

4.5.3 These difficulties would also be encountered should attempts are made to access the 'wisdom of crowds' through crowdsourcing and similar exercises; in this case a covert analysis of communications available from social media, e-mails and other sources may well be the most effective, if more controversial, solution to the problem.

5. Automatic recognition of graph commonalities

5.1 Introduction

5.1.1 It may sometimes happen that separate teams unwittingly analyse the same or similar topics as sub-goals towards the production of comprehensive models of their area of interest. A tool that automatically compared models stored in the database and alerted team members when it seemed that such coincidence of topics had occurred could provide considerable benefit in terms of combination of effort, diversity of background, consistency of output and avoidance of duplication of effort.

5.1.2 The skeleton of how such a tool might work can be laid out at this stage, but fleshing out of the details of its operation would benefit greatly from real world examples of such duplicated nets. Without such examples, and also examples of non-duplicated nets in the same domain, it is nearly impossible to guess which apparent forms of similarity between nets are good evidence of a coincidence of topics, and which occur merely through accident and are not useful.

5.1.3 There is a small set of characteristics of Bayesian networks might be used to recognise that they have essentially common subgraphs: network topology, common or similar node naming, common or similar node value naming and numerically similar probability distributions. It should be recognised that independent projects might model the same subject with different levels of detail, and in particular, parent nodes might be present in one net that are absent in another.

5.2 Algorithm outline

5.2.1 A tool to spot such commonalities would have two algorithmically interesting components. The first would be a function that would take as input a candidate set of correspondences between nodes in the two graphs, and would produce as output a score for how likely it is that the candidate set corresponds to a genuine subject commonality. The second component would take as input a pair of Bayesian networks, and would produce as output all those candidate correspondence sets with a score higher than some threshold.

5.2.2 For the score function, a Bayesian approach is sensible, among other reasons because it allows the fusion of disparate pieces of data in a robust manner, and limits the dimensionality of the algorithm parameter space, thereby enabling effective values for the parameters to be found. The naïve Bayes algorithm has further advantages in terms of reducing the size of the parameter space and increasing algorithm comprehensibility, which is useful for debugging and tuning.

5.2.3 A sensible quantity to use for the score would be the log of the posterior likelihood ratio between the candidate subgraphs having the same underlying subject and their having different subjects. In the naïve Bayes algorithm, this log likelihood is a sum of the log likelihoods for the different features. Here the features are, for each node correspondence, a score for the similarity of the node names, and a sum over all possible node value correspondences of a score for the similarity of the set of node value names multiplied by a score for the similarity of the assigned conditional probabilities. In addition to the per-node score, there would be contributions to the total score from whether or not corresponding pairs of nodes had the same or similar connectivity in the two graphs.

5.2.4 The estimation of the similarity between names could be enhanced by stemming and through the use of a lexical database such as WordNet.

5.2.5 Conceptually, the second component could be thought of as iterating through all subsets of nodes in the first graph and all possible correspondences between those nodes and nodes from the second graph. Such an iteration would not be completed in a reasonable time, even for moderate sized graphs, so a more complex algorithm is required that, so far as possible, would produce the same results.

5.2.6 One possible algorithm for this task is depth first backtracking search, starting by associating a single node from the first graph with a single node from the second graph, and then attempting to extend the correspondence, node by node following links in the first graph, and pruning the search tree when the partial score for the association so far gets too low. Whenever the tree is pruned, the next association (in lexicographic order) is tried. When all the associations for a node have been iterated through, associations not using that node are then analysed. In this way all subsets of the first net are iterated over, or skipped, and for each such subset, all possible associations between it and the second net are iterated over or skipped.

5.2.7 After all the correspondence sets with score above some threshold have been found, they should be ranked in score order and iterated through from the lowest scoring to the highest, deleting those correspondence sets that are either contained in or that contain a set with a higher score.

5.3 Conclusion

5.3.1 The outline of a possible algorithm for discovering common or similar subsets of different Bayesian nets has been described. However, the details of the score functions, and of the algorithm itself that are needed for good performance would depend on the nature of the Bayes nets used and also on the diversity between the way different teams formulate a given subject in Bayes net terms. For this reason, it is suggested that implementation of these ideas in code is postponed until analysts have produced some Bayes nets, ideally using the software tool, and until those nets have been evaluated with this aim in mind.

6. A Bayesian framework for opinion combination

6.1 Introduction

6.1.1 On encountering a problem in the interpretation of observations one's first reaction may well be to seek the advice of others, perhaps more expert than oneself, whose opinions can be canvassed and combined to produce a consensus in which one might place more faith than one would in one's own, or indeed any other single contributor's, individual assessment. This recognition, that two heads may be better than one and the crowd wiser than the individual, has its roots in antiquity and yet remains a topic of current interest and research, As it is proposed that the quantitative probabilistic models that provide the directed links in a Bayes net be constructed on the basis of available expert opinion we should consider how this process of consultation might itself be accommodated within the Bayesian framework. Initially we will discuss the process in rather less personal terms, of combining the outputs of several classifiers.

6.1.2 At first sight the Bayesian Model Averaging (BMA) approach reviewed in [12] seems to be well suited to application to this problem. In this the elements of a set of *K* models (which we identify here with individual classifiers) are assigned prior probabilities $P(k), \quad k = 1 \cdots K$ characterising the degree of belief we associate with their outputs. The parameters characterising these models are determined from (or 'trained on" in the classifier case) a set of data *D*; this process provides us with a set of marginal likelihoods $P(D \mid k)$ obtained if necessary by integration over model/classifier parameters, which can in turn, through Bayes theorem, yield posterior probabilities

$$P(k \mid D) = \frac{P(D \mid k)P(k)}{P(D)} \qquad (6\text{-}1)$$

6.1.3 These can then be used to weight the classifier outputs when these are applied subsequently to test data points x_i to give an averaged prediction of the classification outcome t_i

$$P(t_i \mid x_{i,}, D) = \sum_{k=1}^{K} P(t_i, k \mid x_i, .D) = \sum_{k=1}^{K} P(t_i \mid k, x_i, .D)P(k \mid D) \qquad (6\text{-}2)$$

Here we have assumed that the subsequent data points x_i are independent of the test or training data; the result we obtain takes the form of a sum of predictive classifier distributions, each weighted by the associated classifier's posterior probability.

6.1.4 This procedure sits well within the Bayesian framework, and is widely used in data modelling and prediction [12]. In the current context, however, it does present several problems. Firstly it is only valid if we assume that the *K* classifiers between them capture all the processes whereby the data may be generated mutually exclusively and exhaustively. In practice we may not have such faith in the classifiers' design and performance and yet still wish to make use of their output in the interpretation of available data. The construction, and even the definition, of the marginal likelihood $P(D \mid k)$ can be problematic, whether we consider automatic classifiers or human experts. The latter, in particular, are unlikely to express their opinions in probabilistic terms from which a marginal likelihood could be deduced. Finally we note that the classifiers, and particularly the human experts with their

lifetimes' experiences, need not be trained on the same sets of data or adopt the same prior assumptions. As a result the simple BMA approach would encounter difficulties, assuming as it does that the posterior probabilities $P(k \mid D)$ are conditioned on the same data set *D*.

6.1.5 Nonetheless it is possible to accommodate the contributions of several classifiers, including human experts, within a more realistic framework that is still motivated by Bayesian considerations. This has its origins in the analysis of error rates in medical diagnosis [13], where both the data (symptoms reported by the patient) and its interpretation (in the light of different clinicians' experiences and expertise) are subject to human error. (In [1] it is argued that this medical diagnosis problem might serve as a useful proxy for intelligence-derived problems.) In its original applications the output of this analysis provided useful insights into the data collection process, monitoring the performance of individual clinicians, generating a consensus from the opinions of several clinicians and the possibility of non-expert, and even automated, data collection and interpretation. Subsequently it has been recognised that this framework can provide the basis for a more general description of classifier combination [14], which has recently found application in the analysis of 'crowd sourcing' the interpretation of cosmological data [15].

6.2 Bayesian classifier combination

6.2.1 The development of this Bayesian classifier combination (BCC) approach proceeds in two stages: establishing the principles of its underlying probabilistic model and devising appropriate, and perhaps approximate, numerical methods for its implementation. The component parts of the model are simple and familiar; the interrogated entity is assumed to be in one of *J* classes, labelled by $j = 1, \cdots J$. Thus if the true label associated with the *i*th observation is t_i then

$$\text{Prob}(t_i = j \mid \mathbf{p}) = p(j) \qquad \textit{(6-3)}$$

and

$$\sum_{j=1}^{J} p(j) = 1 \qquad \textit{(6-4)}$$

and **p** is the *J* dimensional vector of components *p*(*j*).

6.2.2 The performance of the *k*th classifier, attributing the observation of the *i*th observation to one of the same *J* classes, labelled now by $c_i^{(k)}$, is characterised by a so-called confusion matrix such that

$$\text{Prob}\left(c_i^{(k)} = j\right) = \pi_{j,c_i^{(k)}}^{(k)} \qquad \textit{(6-5)}$$

with

$$\sum_{c_i^{(k)}=1}^{J} \pi_{j,c_i^{(k)}}^{(k)} = 1 \qquad \textit{(6-6)}$$

as the classifier is certain to respond to an observation with an assignment to one of the *J* classes.

6.2.3 It is reasonable to assume that we know the classifier outputs $c_i^{(k)}$, but do not know either the true observed state t_i, the $p(j)$ or the confusion matrices $\pi_{j,c_i^{(k)}}^{(k)}$. From these we construct the joint probability of $\mathbf{c}_i \equiv \left\{c_i^{(k)}, k = 1,\cdots K\right\}$, t_i as

$$P(\mathbf{c}_i, t_i \mid \mathbf{p}, \boldsymbol{\pi}) = p(t_i)\prod_{k=1}^{K} \pi_{t_i,c_i^{(k)}}^{(k)} \qquad (6\text{-}7)$$

6.2.4 If we treat the t_i as hidden variables, **p** and **π** as parameters and **c** as input, we can now attempt to find **p** and **π** by likelihood maximisation. To do this we must assign prior distributions to **p** and **π** that are in turn specified by appropriate hyper-parameters. Thus we assume that the components of **p** have the Dirichlet distribution

$$P(\mathbf{p} \mid \mathbf{v}) = \Gamma\left(\sum_{j=1}^{J} v_j\right)\prod_{j=1}^{J} \frac{p(j)^{v_j-1}}{\Gamma(v_j)}\delta\left(1-\sum_{j=1}^{J} p(j)\right) \qquad (6\text{-}8)$$

with hyper-parameter $\mathbf{v} \equiv \left\{v_j, j = 1,\cdots J\right\}$ and Γ denoting the gamma function; the delta function imposes the condition (2-4). Some properties of this distribution are reviewed in an appendix of [1], as well as in the texts [2] and [3]. The prior associated with **π** takes a similar form, with the row of elements $\left\{\pi_{j,1}^{(k)} \cdots \pi_{j,J}^{(k)}\right\}$ with a Dirichlet PDF characterised by the hyper-parameters $\alpha_{j,1}^{(k)} \cdots \alpha_{j,J}^{(k)}$ through $P(\boldsymbol{\pi} \mid \boldsymbol{\alpha})$.

6.2.5 In this way we can now set

$$P(\mathbf{t}, \mathbf{p}, \boldsymbol{\pi} \mid \mathbf{c}.) \propto \prod_{i=1}^{N}\left\{p(t_i)\prod_{k=1}^{K} \pi_{t_i,c_i^{(k)}}^{(k)}\right\} P(\mathbf{p} \mid \mathbf{v})P(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) \qquad (6\text{-}9)$$

as a consequence of Bayes theorem. Using this model form of the joint PDF and the classifier outputs **c** we must now attempt to determine the parameters **p** and **π**. This problem can be addressed by seeking to maximise the model PDF, given **c**, by varying the **p** and **π** .and integrating over the hidden variables **t**. This maximisation is perhaps best performed iteratively using an expectation - maximisation (EM) algorithm or the closely related variational Bayes (VB) approach.

6.3 Expectation-maximisation and variational Bayes methods

6.3.1 The iterative EM approach to maximum likelihood estimation was introduced by Dempster et al [16] and its principles clarified and set in a broader context by Neal and Hinton [17]. Chapters 9 and 10 of [2] give a detailed and accessible account of these methods, illustrating their principles through concrete examples based on Gaussian mixture and other models. Here we will briefly review the VB approach adopted in [15] as this yields a computationally efficient and effective algorithm and presents the underlying principles in a form that allows us to consider ways in which the simplifications made to render this model tractable might be relaxed, albeit at a considerable cost in computational overhead. To motivate our discussion, however, we first consider the case where the observed variables **X** and hidden variables **Z** have a joint PDF $P(\mathbf{X},\mathbf{Z})$ with the marginal PDF of **X** being written in terms of the conditional PDF of **Z**, given the value of **X**, as

$$P(\mathbf{X}) = \frac{P(\mathbf{X},\mathbf{Z})}{P(\mathbf{Z} \mid \mathbf{X})} \qquad (6\text{-}10)$$

6.3.2 If we require information about the hidden variables **Z**, given our observations **X,** this will be contained in the function $P(\mathbf{Z} \mid \mathbf{X})$, which we hope to approximate by the function $q(\mathbf{Z})$; as this is a PDF, it must satisfy the normalisation condition

$$\int q(\mathbf{Z}) d\mathbf{Z} = 1 \qquad (6\text{-}11)$$

6.3.3 Thus, if we take the logarithms of both sides of (2-10), pre-multiply each by $q(\mathbf{Z})$ and integrate over **Z**, we find that

$$\begin{aligned} \log(P(\mathbf{X})) &= \int q(\mathbf{Z})[\log(P(\mathbf{X},\mathbf{Z})) - \log(P(\mathbf{Z} \mid \mathbf{X}))] d\mathbf{Z} \\ &= \int q(\mathbf{Z}) \log\left(\frac{P(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})}\right) d\mathbf{Z} - \int q(\mathbf{Z}) \log\left(\frac{P(\mathbf{Z} \mid \mathbf{X})}{q(\mathbf{Z})}\right) d\mathbf{Z} \\ &= \Lambda(q) + \mathrm{KL}(q(\mathbf{Z}) \mid P(\mathbf{Z} \mid \mathbf{X})) \end{aligned} \qquad (6\text{-}12)$$

Here $\mathrm{KL}(q(\mathbf{Z}) \mid P(\mathbf{Z} \mid \mathbf{X}))$ is the Kullback Liebler divergence of the PDFs $q(\mathbf{Z})$ and $P(\mathbf{Z} \mid \mathbf{X})$, which, as is discussed in [2] and [3], is necessarily greater than or equal to zero, and vanishes only when these two functions are identical. (These arguments, based on the analysis of Neal and Hinton [17] follow those of [2] chapter 10 closely, and correct a sign error in equation (2) of [15]) As *P*(**X**) is independent of **Z**, and so cannot change as *q* is varied, the functional

$$\Lambda(q) = \int q(\mathbf{Z}) \log\left(\frac{P(\mathbf{X},\mathbf{Z})}{q(\mathbf{Z})}\right) d\mathbf{Z} \qquad (6\text{-}13)$$

will take its maximum value when $q(\mathbf{Z})$ and $P(\mathbf{Z} \mid \mathbf{X})$ are the same. We note that the joint PDF $P(\mathbf{X},\mathbf{Z})$ (and its logarithm) are relatively simple in the case of our model (2-9); in this case the minimisation of $\Lambda(q)$ provides a potentially convenient route to a useful approximation to $P(\mathbf{Z} \mid \mathbf{X})$.

6.3.4 Obviously the functional form of $q(\mathbf{Z})$ will have to be very flexible and carefully controlled if this approach is expected to yield a very realistic caricature of $P(\mathbf{Z} \mid \mathbf{X})$; the fidelity of the representation q provides is invariably compromised by the requirement that the calculations involved in the search for the maximum in $\Lambda(q)$ are tractable. There are significant parallels between the construction of such tractable forms for $q(\mathbf{Z})$ and approximations employed in the classical statistical mechanics of interacting spin systems. One of the most commonly made of these is the so-called mean field approximation, whose description of the assembly of spins in terms of independent individual spins interacting with their fellows through an average or 'mean' field is analogous to the factorization

$$q(\mathbf{Z}) = \prod_i q_i(\mathbf{z}_i) \qquad \textit{(6-14)}$$

where $\mathbf{z}_i$ are the individual elements making up the set of hidden variables. Yedidia et al. [18] review these analogies in some detail and shows that the message passing or sum/product algorithms discussed in [1] are equivalent to a well-known extension of the mean field theory model due to Bethe and Peierls [19].

6.3.5 On introducing the product form (2-14) of $q(\mathbf{Z})$ into (2-13) we find that

$$\Lambda(q) = \int \left(\prod_i q_i(\mathbf{z}_i) d\mathbf{z}_i \right) \left[\log(P(\mathbf{X}, \mathbf{Z})) - \sum_i \log(q_i(\mathbf{z}_i)) \right] \qquad \textit{(6-15)}$$

If we now consider the maximisation of $\Lambda(q)$ with respect to variations in just one of the $q_i(\mathbf{z}_i)$ we re-write this as

$$\Lambda(q_i) = \int d\mathbf{z}_i q(\mathbf{z}_i) [\log(F_i(\mathbf{z}_i)) - \log(q(\mathbf{z}_i))] + C_i \qquad \textit{(6-16)}$$

and C_i incorporates all terms insensitive to variations in $q_i(\mathbf{z}_i)$. The first term in (2-16) can be recognised as minus the divergence $\mathrm{KL}(q_i(\mathbf{z}_i) \mid F_i(\mathbf{z}_i))$ which will vanish (and $\Lambda(q_i)$ will take its maximum value) when we identify the optimum form of $q_i(\mathbf{z}_i)$ (denoted by an asterisk) as

$$\log q^*_i(\mathbf{z}_i) = \log(F_i(\mathbf{z}_i))_i \qquad \textit{(6-17)}$$

This result can also be written as

$$\log q^*_i(\mathbf{z}_i) = \mathrm{E}_{j \neq i}(\log(P(\mathbf{X}, \mathbf{Z})))_i \qquad \textit{(6-18)}$$

where the expectation value is taken over all the $\mathbf{z}_j$ except for $\mathbf{z}_i$.So, once the observed data $\mathbf{X}$ are known, the factors $q_i(\mathbf{z}_i)$ can be initiated, then updated in turn by evaluating the expectations in (2-18) on the basis of the current values of the other $q_j(\mathbf{z}_j)$. This can be iterated until reasonable convergence has been achieved and the resulting $q^*_i(\mathbf{z}_i)$ used to characterise the hidden variables.

6.3.6 In the case of the model developed in section 2.2 considerable progress can be made in analytic terms, greatly reducing the associated computational burden. The averaging over the **t**, **p** and **π** variables is facilitated by the introducing the following notation

$$N_j = \sum_{i=1}^{N} q^*(t_i = j) \qquad (6\text{-}19)$$

$$N_{jl}^{(k)} = \sum_{i=1}^{N} \delta_{c_l^{(k)},l}\, q^*(t_i = j)$$

where $\delta_{m,n} = 1$, when $m = n$, and 0 otherwise and recalling that

$$\mathrm{E}_{\mathbf{p}}[p(k)] = \Gamma\left(\sum_{j=1}^{J} \nu_j\right) \int \log(p(k)) \delta\left(1 - \sum_{j=1}^{J} p(j)\right) \prod_{j=1}^{J} \left(\frac{p(j)^{\nu_j - 1}}{\Gamma(\nu_j)} dp(j)\right) \qquad (6\text{-}20)$$

$$= \Gamma\left(\sum_{j=1}^{J} \nu_j\right) \prod_{j=1}^{J} \left(\frac{1}{\Gamma(\nu_j)}\right) \frac{\partial}{\partial \nu_k} \left[\frac{\prod_{j=1}^{J} (\Gamma(\nu_j))}{\Gamma\left(\sum_{j=1}^{J} \nu_j\right)}\right] \quad 1 \le k \le J$$

$$= \psi(\nu_k) - \psi\left(\sum_{j=1}^{J} \nu_j\right)$$

Here ψ is the logarithmic derivative of the gamma function [20]

$$\psi(z) = \frac{1}{\Gamma(z)} \frac{d\Gamma(z)}{dz} \qquad (6\text{-}21)$$

Using these results it can be shown that (c.f. (2-18))

$$\log q^*(t_i) = \mathrm{E}_{\mathbf{p}}[\log(p_{t_i})] + \sum_{k=1}^{K} \mathrm{E}_{\boldsymbol{\pi}}\left[\log\left(\pi_{t_i, c_i^{(k)}}^{(k)}\right)\right] + C \qquad (6\text{-}22)$$

where C is a normalisation constant manifest in the logarithmic domain. Thus, if we initiate the parameters $\mathbf{v}, \boldsymbol{\alpha}$ as $\mathbf{v}_0, \boldsymbol{\alpha}_0$ we can use these to evaluate $\log q^*(t_i)$ from (2-20) and (2-22). These values are then used to evaluate the 'populations' (2-19). We now see that (again much as in (2-18)) that

$$\log q^*(\mathbf{p}) = \sum_j N_j \log(p(j)) + \log(P(\mathbf{p} \mid \mathbf{v}_0)) + C \qquad (6\text{-}23)$$

$$= \sum_j (N_j + \nu_{0j} - 1) \log(p(j)) + C$$

which implies the parameter update

$$\nu_j = \nu_{0j} + N_j \qquad (6\text{-}24)$$

The $\boldsymbol{\alpha}$ parameters are then reset analogously to

$$\alpha_{jl}^{(k)} = \alpha_{0jl}^{(k)} + N_{jl}^{(k)} \qquad (6\text{-}25)$$

and the process can be iterated until convergence is achieved. The resulting PDF (2-9) can then be interpreted in terms of the inferred values of **t** with probabilities characterised by **p**, while the performances of the various classifiers will be reflected in the values taken by the parameters $\boldsymbol{\pi}$.

6.4 Some extensions and applications of the BCC

6.4.1 The analysis of the Bayesian classifier combination in section 2.2 and 2.3 has been carried through to yield computationally tractable results as a consequence of both the simplifying assumptions implicit in its formulation and the approximate, variational Bayes, method used in its solution. Here we will consider briefly both how these assumptions and approximations might be relaxed and some applications of the model made feasible by its computational simplicity.

6.4.2 The account [14] of the BCC addresses both these issues. While its initial formulation of the model is very similar to that described in section 2.2, the method of solution it proposes is quite different. Rather than initialize the parameters $\boldsymbol{\alpha}$ with specific values these are ascribed exponential prior distributions that are in turn chosen to give a significant weighting to the diagonal elements of the confusion matrices, and so incorporate the reasonable supposition that the classifiers perform significantly better than a random assignment to classes. The implementation of the EM algorithm used to determine these parameters by likelihood maximization adopts a Gibbs sampling based approach (see chapter 11 of [2]) similar to that described in [13]. This greatly increases the computational burden associated with its solution.

6.4.3 The assumption that the classifier outputs are independent can be relaxed in several ways. The simplest of these described in [14] assumes that some of the classifications are ‘easy’ and are performed subject to a preset confusion matrix with small off diagonal elements; the remaining classifications are taken to be ‘hard’ and their implementation is modeled through the $\boldsymbol{\pi}$ variable as before; the assignment of data points to these ‘hard’ and ‘easy’ classes is made randomly by ‘the toss of a coin’ manifest through a Bernoulli latent variable. The inference based on this so-called enhanced BCC model is again carried through using Gibbs sampling. Alternatively the classifier outputs can be modeled by a Markov network which captures both their cross correlations and correlations with the hidden variables **t** and replaces the confusion matrices $\boldsymbol{\pi}$. Preliminary tests of these models, undertaken on data sets containing DNA and satellite signature data and described in [14], were rather inconclusive; the independent and enhanced BCC methods yielded similar results, while the Markov network approach worked reasonably well on some data sets, but not on others. [15] also presents a comparison of the VB and Gibbs sampling solutions of the independent BCC; for the data sets they considered the former appeared to perform significantly better.

6.4.4 The much greater computational efficiency of the VB BCC method allows it to be applied to extensive data sets and as tool with which the pool of human base classifiers can be investigated. In the example considered in [5] a cohort of civilian scientists assessed astronomical data to classify events as being very likely or not at all likely to have been derived from a supernova event; a third classification, of the scientist not feeling confident to make any assessment, was also available, Post-hoc verification of supernova events on the basis of spectroscopic data provided an independent test of the inferred conclusions. In addition to out-performing other methods of classifier combination the VB BCC analysis was able to extract significant and sensible structures from the pool of human analysts. In particular the $\boldsymbol{\pi}$ matrices were able to identify effective and ineffective base classifiers and as a basis for

subsequent re-assignment to other tasks. The dynamics of the evolving expertise of the group of analysts was also accessible, and was modeled quite effectively using a dynamical model analogous to that exploited in a Kalman filter. Finally we mention a very recent application of the VB BCC method to the analysis, similar to that mentioned in section 1.4.3, of sentiment derived from text streams, and correlated with the performance of the Non-farmer Payrolls economic indicator [21].

6.5 Conclusions

6.5.1 In this section of the report we have reviewed how the process of combination of opinions might be considered within a Bayesian context While Bayesian model averaging is a useful tool for the analysis of controlled data and its exploitation in predictions, it encounters problems when applied in situations where opinions are formed by human classifiers of different levels of expertise and experience. A method originally developed to assess errors in clinical diagnosis has been extended to produce a framework in which the Bayesian combination of classifiers can be carried out.

6.5.2 By representing the performance of the individual classifiers/experts in terms of the familiar confusion matrix and simplifying the analysis through assumptions of independence of opinions and the adoption of tractable Dirichlet priors it is possible to formulate this framework in such a way that it can be analysed effectively using an approximated variational procedure in a way that, as well as inferring useful information from incomplete and uncertain data, also provides an assessment of the performance of the classifiers whose opinions are being combined. Preliminary applications of this methodology described in the literature have demonstrated its effectiveness in the processing of large quantities of crowdsourced cosmological analysis and identifying signatures of sentiment in text streams that provide an indicator of economic performance. While this approach is for the moment a topic of active research it may well provide a useful tool for the analysis of intelligence problems in the future.

7. Discussion of novel software aspirations

7.1 Introduction

7.1.1 Bayesian networks and the main algorithms used to analyse them were invented by Judea Pearl in the 1980s. As a technology, software systems to assist with the analysis of Bayesian networks are now moderately mature, and have standardised, to some extent, on a set of capabilities that work well as a compromise between what is useful and what is possible, exploiting commonly occurring structure in real-world joint probability distributions and appealing to the intuitions of application domain experts.

7.1.2 Example problems provided by intelligence experts, and discussions with those experts have highlighted that certain extensions to the capability of standard Bayesian network tools could potentially increase their usefulness in this domain. This is, of course, provided that they firstly, are possible and secondly, that the disadvantages from concomitant complications do not outweigh the original benefit.

7.1.3 It was recognised that the novel technology required for such extensions would have to be designed and its implications discussed before a decision was made to implement. This section provides an initial analysis towards those ends, discussing how such technology might work, and how it might affect the user experience.

7.2 Probability ranges

7.2.1 According to both objectivist and subjectivist interpretations of Bayesian probability theory, the useful knowledge about a single proposition can be encoded in a single probability distribution. For example, if the proposition is that a certain country possesses chemical weapons, the useful part of an analyst's knowledge about that proposition can be encoded in a single probability: a number between zero and one inclusive.

7.2.2 In spite of the theoretical support for Bayesian based decision making and the proven effectiveness of Bayesian techniques in a wide range of artificial intelligence type problems, human intuition rebels at the idea that such knowledge can captured in a single number, and if a person is asked what the number is, he is likely to prevaricate and respond with a range of probabilities rather than a single value. How can we explain this apparent disparity between a successful theory and the intuition of humans: arguably the greatest decision maker in existence?

7.2.3 An important piece of evidence bearing on this paradox is that, in the right circumstances, a person will choose an action based on their belief, and deductions about their belief can be made from their choice. In our example, if a person was told that that the truth about the country's chemical weapons will be discovered in a month, and that they have to make a small bet now on whether or not such weapons are found, then their choice of bet at their offered odds can be used to place an upper or lower bound on their opinion of the probability.

7.2.4 A more sophisticated question to elicit the person's opinion of the probability is to ask them to choose a number p between 0 and 1 on the understanding that if the country is discovered to have chemical weapons they will be given £$(1-(1-p)^2)$, and if the country is discovered not to have chemical weapons, they will be given £$(1-p^2)$. For example, they could be asked to choose a row from the following table

p	Reward if CW	Reward if no CW
0.0	£0	£1
0.1	£0.19	£0.99
0.2	£0.36	£0.96
0.3	£0.51	£0.91
0.4	£0.64	£0.84
0.5	£0.75	£0.75
0.6	£0.84	£0.64
0.7	£0.91	£0.51
0.8	£0.96	£0.36
0.9	£0.99	£0.19
1.0	£1	£0

7.2.5 A person might be quite content to play this game, and hence inadvertently divulge their opinion of the probability that the country has chemical weapons although the same person is quite likely to be unhappy to baldly state what he thinks that probability is.

7.2.6 We argue here that a person will be unhappy to state a probability if he thinks it is likely that further evidence is likely to appear that might cause him to adjust his probability (and his game strategy). This is because such an adjustment would feel like making a correction: as though he had been proven wrong. There might also be justifiable concern about unfair reputation loss in the event of adjusting probabilities.

7.2.7 For example, if evidence later comes to light that the country of interest has been cooperating with a second country known to have chemical weapons, then the person might wish to increase his probability. If the appearance of such evidence has not yet happened, but is quite likely to happen, then the person is unlikely to be happy to state a probability now for fear of being 'proven wrong'. Another case in which the person might feel unsure of the stability of his estimate is if information is likely to come to light about a similar country to the country of interest that might be supposed to be operating in the same environment.

7.2.8 In an ideal world, an expanded Bayesian network could be constructed capturing the possible future evidence that might be discovered and its relation to the questions of interest. In the cases where that is possible, we believe that the person would be more comfortable giving un-ranged probability distributions for each node and allowing the net to do the work of adjusting the probabilities of interest in the light of evidence. However, it is likely to be the case that the forms and power of future evidence not known and/or too numerous to list.

7.2.9 We consider three different approaches to accommodating probability ranges in a software tool. The first approach is to allow the user to input a probability range, but then to simply act as though the user had input the center point of that range. This approach might ease the concerns of the user and increase the chance that he use the tool, but the tool might not then behave as the user expects or reflect his intuitions. In particular, the effective value of this probability will not adjust within the specified range to reflect evidence.

7.2.10 A second approach to accommodating probability ranges is to calculate the range of possible answers for the resulting marginal distributions corresponding to the input ranges given for the conditional distributions. This approach has a number of issues: with specifying the ranges, calculating the results, communicating the results to the user and whether the result ranges correspond to what a user would expect.

7.2.11 For a binary variable, it would be straightforward for a user to give a minimum and maximum value for the probability that the probability takes a certain value. For variables that take more than two values however, the situation is more complicated and there are interactions between the allowed ranges of the probabilities for different values.

7.2.12 Interval arithmetic is a computational technique for putting bounds on rounding errors and measurement errors in mathematical computation. It defines new arithmetical operators such as addition, subtraction, multiplication etc. corresponding to the well-known operators, but taking intervals as inputs and producing intervals as outputs. In this way complex computations can be converted to an interval form. Unfortunately, interval arithmetic is subject to an issue called the dependency problem, which leads to the answer interval produced by a calculation being larger than the possible range of values taken by solutions to the corresponding non-interval problem. The particular calculations needed to determine marginal distributions for Bayes nets would be significantly degraded by this dependency problem.

7.2.13 An alternative would be to sample, randomly or uniformly, from the input intervals and to put intervals around the resulting outputs. This would only give approximate results, and would be likely to be slow, but if the underlying Bayes net algorithm were changed to a message passing scheme, it could be workable.

7.2.14 For a binary random variable, an interval result is easy to interpret, but for a random variable with more possible values, taking the range of probabilities for each value throws away information.

7.2.15 The interval approach does not allow the specified probabilities to reflect the evidence, and so the output ranges are likely to include probabilities that would not reflect a user's intuition. An example might make this clearer. In a pub game a man tosses a coin repeatedly, and people bet on the outcome. Suppose that initially it is known that the coin is probably fair, but that it may be biased towards heads, and that the probability of getting a head is 0.5 or 0.9 depending on the coin. If the coin produces a sequence of heads, a player is likely to judge that the coin is biased and think the next toss will probably be a heads. However, if a Bayesian net is used for the analysis, and the range [0.5,0.9] is used for the probability of heads, then even though many heads have been observed, the probability for the next toss will still be given as the range [0.5,0.9].

7.2.16 A third approach to accommodating probability ranges is the use of hyper-prior distributions for the probabilities. These can be thought of as additional nodes inserted into the Bayesian net above nodes with probability ranges, but it is not necessary for these additional nodes to be displayed to the user. The effective value of a probability specified in this way would update in an intuitive manner. In the coin tossing example, a hyper-prior would allow the network to gradually come to 'realise' that the coin was biased, and adapt its predictions for the next toss accordingly.

7.2.17 In order that the net calculations be tractable, it would probably be necessary for the hyper-prior distributions to be either discrete or Dirichlet. These may not correspond to quite what an analyst means by specifying a probability range, but we believe that even though it is just an approximation, the behaviour from a Dirichlet hyper-prior will match an analyst's intuition fairly well, and integrate observations in a robust and sensible manner. A user could input a mean probability distribution, and adjust a parameter encoding how certain he is while viewing the resulting (2 sd) ranges for each value of the variable. In the case of a binary variable, the user could input a range instead.

7.2.18 Incorporating this third approach into the software would probably take a few weeks.

7.3 Allowing undefined conditionals and the presence of zeroes

7.3.1 The joint probability distribution for the 'RedCW' example problem can be represented as a product $p(CW,C,U) = p(CW)p(C \mid CW)p(U \mid C,CW)$. This representation corresponds to a Bayesian net with arrows from CW to C and U, and from CW to U. Since all three variables are binary, seven conditional distributions must, in general, be specified in order to define the joint, namely: p(CW), p(C|CW), p(C|-CW), p(U|C,CW), p(U|C,-CW), p(U|-C,CW) and p(U|-C,-CW). However, in the problem statement one of these: p(C|-CW) is not defined, but because p(U|C,-CW) and p(U|-C,-CW) are both defined to be zero, the joint distribution can still be deduced. This is because the unknown quantities are multiplied by zero in the expression for the unnormalised joint distribution.

7.3.2 A related issue occurs with the Cards problem. In that problem, no probabilities are specified, but there is an implicit assumption that all combinations of letter types and number types are possible. The knowledge that all the conditional probabilities for the card faces are non-zero is sufficient to deduce which cards need to be turned over (i.e. which hidden faces have non-zero mutual information with the all-cards-valid node).

7.3.3 These use cases could be accommodated in the software by allowing conditional probabilities to be undefined, or defined only as non-zero. The computation of marginals and mutual information (subject to observations) would be modified to encode that zero multiplied by anything is zero and that two non-zeroes multiplied together give a non-zero. The result of the computation of a marginal or a mutual information might now take one of the additional values 'non-zero' or 'unknown'.

7.3.4 Incorporating these augmentations to the current software would probably take a few weeks. We have not yet thought through in detail how well these augmentations would combine with the hyper-prior work, but our investigations so far suggest that they could be combined. In the case of one of these non-numerical answers, the non-numerical inputs that caused it could be listed.

7.3.5 One caveat with both these augmentations is that it is not clear that a message passing algorithm would work in conjunction with them. At the moment, an exhaustive algorithm is used, and more efficient exact algorithms could be substituted if necessary, but beyond a certain complexity of network, radically different algorithms such as message passing will need to be used for efficiency reasons, and it is not clear that these could be combined with the suggested software augmentations.

7.4 Arbitrary analyst specifications: under and over determination and ill conditioning

7.4.1 With standard Bayesian net software, as with the current version of Banish, the user has to supply a conditional probability distribution for each node, for each possible combination of values of its parent nodes. This set of information is automatically self-consistent, consistent with the conditional independence information specified by the network topology and, when combined with the conditional independence information, is complete. Furthermore, this information is naturally well-conditioned in the sense that small proportionate changes to the inputs lead to small changes to the outputs.

7.4.2 It would be advantageous to make a tool that could accept more general quantities associated with the joint distribution than the conditional distributions described in the previous paragraph. Also, different pieces of information might come from different analysts. It would be useful if the software could spot when the information given to it

was inconsistent, badly conditioned, or incomplete, and in the case that it's incomplete, if it could suggest what information would be needed to complete it.

7.4.3 In the most general case, a joint probability in n variables is represented by a complete Bayes net. That is to say, every pair of nodes is connected. In this case, the number of scalar quantities needed to specify the joint distribution is one less than the product of the number of values of each of the nodes.

7.4.4 For example, the RedCW net has three nodes: CW, Confict and Use, and all three are binary nodes. There are arrows from CW to Conflict and Use, and an arrow from Conflict to Use. There are therefore eight possible combinations of values for these variables, and since the probabilities of these combinations must sum to 1, 7 different probabilities need to be specified in order to completely specify the joint distribution.

7.4.5 A user might specify the probability of any subset of the eight combinations. There are 256 such subsets, though the probability of the empty subset and the complete subset are necessarily 0 and 1 respectively. In general, a user might specify the joint distribution by specifying the probabilities of 7 subsets of the remaining 254. There are 254C7 ~= 1.2e13 possible choices for the 7, and a significant proportion of these are independent.

7.4.6 Each specification of the probability of a subset can be thought of as a linear relation on the probabilities of the eight combinations. The tools of linear algebra can be used to determine if these relations (together with the summing to one condition) are independent and span the whole space (i.e. whether the specifications are consistent and complete), and to find the solution satisfying the relations (each component of which should be positive). Linear algebra tools would also allow the conditioning of the specification to be calculated. If the relations do not span the whole space, then most of the remaining relations from the set of 254 could be used to increase them to a spanning set. Although all such elements from the 254 could be suggested to the user, it would be too big a set to be useful to the user, and for a bigger Bayesian net, the set of suggestions would be much bigger even than that. For example for a four node net, there are about 65 thousand relations.

7.4.7 The complete network structure on the nodes specifies an ordering of the nodes: CW, Conflict, Use. If the full joint distribution is defined, then so are the joint distributions on the sets {CW}, {CW, Conflict} and {CW, Conflict, Use}. If the specified values are not complete, a plausible strategy for specifying which relations should be requested from the user is to request relations to define the conditional distributions for each of the nodes in turn, conditioned on the previous nodes from the sequence. It is straight-forward to iteratively find which relations should be requested given those already specified, and there would only be a manageable number of requests for a reasonably sized net.

7.4.8 Although this description has concentrated on the RedCW network, it applies to any complete (i.e. fully connected) network. For an incomplete network things are more complicated. Missing arrows in the graph correspond to conditional independence relations between random variables. Bayesian nets are so useful because naturally occurring probability distributions tend to have conditional independence properties and because such properties greatly reduce the scale of the task of specifying a joint probability distribution.

7.4.9 Although the knowledge of the conditional independence relations between the nodes of a net could be discarded before the consistency/completeness analysis was performed, the resulting system would be very onerous to use because all the discarded information would have to be relearned by the system through information requests from the user. This would consume a lot of user time and make unrealistic demands on the user matching the conditional independence relations with the

numbers he puts in. On the other hand, the conditional independence relations, considered as relations between the probabilities of the value combinations are non-linear, and including them with the specified set of relations would lead to a problem in algebraic geometry. The solution to such a problem is not likely to be useful: for instance, it might be a finite set of points in probability space, or perhaps parabaloid of revolution. It is not clear in this case how the user could be prompted for further information that would not contradict what was already known.

7.4.10 A possible compromise between discarding all the conditional independence information and keeping it all would be to group together those nodes that are in a loop in the original graph, and add edges to make the corresponding subgraph complete. The graph would then have the form of a tree on the large scale, where a node of the tree corresponded to a complete subgraph. Questions of consistency, completeness, conditioning and the requesting of new relations could then be solved for the complete subgraphs separately and starting for earlier subgraphs in the tree.

This section describes highly experimental ideas. Although it is almost certainly practical to implement these ideas, it is not at all clear that the resulting system would be useful in a practical context. Trying to combine these ideas with the other suggested augmentations would be premature at this stage. If there is interest in pursuing this line of investigation, production of a stand-alone tool would be a sensible (and non-trivial) first step.

8. Customer Feedback

8.1.1 As part of our commitment to our customers, and to the IS0 9001:2008 quality standard, we would welcome feedback on any aspect of our work, good or bad. If you have any comments or queries regarding this report, please send your enquiries to radar@igence.com.

9. References

(1) T M Cooper, R D Hill and R J A Tough, "*ISTAR modelling interim report*", IR0001054, December 2013

(2) C M Bishop, 'Pattern Recognition and Machine Learning', Springer Verlag, New York, 2006

(3) D J C MacKay, "Information theory, inference and learning algorithms", University Press, Cambridge, 2003

(4) J Zlotnick, '*Bayes theorem for intelligence analysts*', Studies in Intelligence, **16**, No. 2, 1972

(5) N Schweitzer, '*Bayesian analysis for intelligence: some focus on the Middle East*', Studies in Intelligence, **20**, No. 7, 1976

(6) N Dalkey and O Helmer, '*An experimental application of the Delphi method to the use of experts*', Management Science, **9**, 458-467, 1963

(7) H A Linstone and M Turoff, '*The Delphi method: techniques and applications*', Addison Wesley, New York, 1975

(8) http://polymathprojects.org/

(9) J Surowiecki, 'The Wisdom of Crowds: Why the Many are Smarter than the Few', Doubleday, New York, 2004

(10) D C Brabham, '*Crowdsoucing as a model for problem solving*', Convergence: The International Journal of Research into New Media Technologies, **41,** 75-90, 2008

(11) R P Schumaker and H Chen, "*Textual analysis of stock market prediction using breaking financial news: The azfin text system*," ACM Trans. Inf. Syst., **27**, no. 2, pp. 12:1–12:19, 2009.

(12) J A Hoeting, D Madigan, A E Raftery and C T Volinsky, '*Bayesian Model Averaging :A Tutorial*', Stat. Sci., **14**,382-417, 1999

(13) A P Dawid and A M Skene, '*Maximum likelihood estimation of observer error rates using the EM algorithm*', J. Roy. Stat. Soc., Series C, **28**, 20-28, 1979

(14) Z Ghahramani and H C Kim, '*Bayesian classifier combination*', Gatsby Computational Neuroscience Unit Technical Report GCNU_T, London, UK, 2003

(15) E Simpson, S Roberts, I Psorakis and A Smith, '*Dynamic Bayesian combination of multiple imperfect classifiers*', arXiv:1206.1831v1 [math ST], 8 Jun 2012

(16) A P Dempster, N M Laird and D B Rubin, '*Maximum likelihood from incomplete data via the EM algorithm*', J. Roy. Stat. Soc., Series B, **39**, 1-38, 1977

(17) R M Neal and G Hinton, '*A view of the EM algorithm that justifies incremental, sparse and other variants*', in M I Jordan (Ed.), 'Learning in Graphical Models', 355-368, MIT Press, 1999

(18) J.S. Yedidia, W. T. Freeman and Y. Weiss, '*Understanding belief propagation and its generalisations*', Chap. 8, 'Exploring Artificial Intelligence in the New

Millennium', G. Lakemeyer and B. Nebel (Eds.), Morgan Kaufmann, San Francisco, 2003

(19)K. Huang, 'Statistical Mechanics', Sec. 16.5, John Wiley, New York, 1963

(20) F W J Olver, D W Lozier, R F Boisvert and C W Clark (Eds.), 'NIST Handbook of Mathematical Functions', Chap. 5, University Press, Cambridge, 2010

(21)A Levenberg, S Pulman, K Moilanen, E Simpson and S Roberts, '*Predicting economic indicators from web text using sentiment composition*'. Proc. 3rd International Conference on Information Computer Applications, Barcelona, 2014