Towards a Noise-Tolerant, Representation-Independent Mechanism for Argument Interpretation[*]

Ingrid Zukerman and **Sarah George**
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800, AUSTRALIA

Abstract

We describe a mechanism for the interpretation of arguments, which can cope with noisy conditions in terms of wording, beliefs and argument structure. This is achieved through the application of the Minimum Message Length Principle to evaluate candidate interpretations. Our system receives as input a quasi-Natural Language argument, where propositions are presented in English, and generates an interpretation of the argument in the form of a Bayesian network (BN). Performance was evaluated by distorting the system's arguments (generated from a BN) and feeding them to the system for interpretation. In 75% of the cases, the interpretations produced by the system matched precisely or almost-precisely the representation of the original arguments.

1 Introduction

In this paper, we focus on the interpretation of argumentative discourse, which is composed of implications. We present a mechanism for the interpretation of NL arguments which is based on the application of the Minimum Message Length (MML) Principle for the evaluation of candidate interpretations (Wallace and Boulton, 1968). The MML principle provides a uniform and incremental framework for combining the uncertainty arising from different stages of the interpretation process. This enables our mechanism to cope with noisy input in terms of wording, beliefs and argument structure, and to factor out the elements of an interpretation which rely on a particular knowledge representation.

So far, our mechanism has been tested on one knowledge representation – Bayesian Networks (BNs) (Pearl, 1988); logic-based representations will be tested in the future. Figure 1(a) shows a simple argument, and Figure 1(d) shows a subset of a BN which contains the preferred interpretation of this argument (the nodes corresponding to the original argument are shaded). In this example, the argument is obtained through a web interface (the uncertainty value of the consequent is entered using a drop-down menu). As seen in this example, the input argument differs structurally from the system's interpretation. In addition, the belief value for the consequent differs from that in the domain BN, and the wording of the statements differs from the canonical wording of the BN nodes. Still, the system found a reasonable interpretation in the context of its domain model.

The results obtained in this informal trial are validated by our automated evaluation. This evaluation, which assesses baseline performance, consists of passing distorted versions of the system's arguments back to the system for interpretation. In 75% of the cases, the interpretations produced by the system matched the original arguments (in BN form) precisely or almost-precisely.

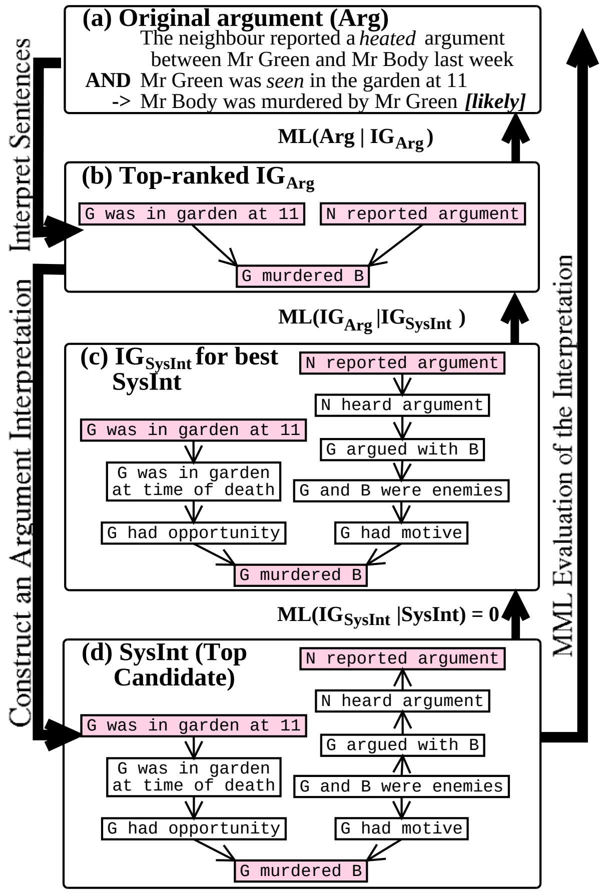
In the next section, we review related research. We then describe the application of the MML criterion to the evaluation of interpretations. In Section 4, we outline the argument interpretation process. The results of our evaluation are reported in Section 5, followed by concluding remarks.

2 Related Research

Our research integrates reasoning under uncertainty for plan recognition in discourse understanding with the application of the MML principle (Wallace and Boulton, 1968).

BNs in particular have been used in several such plan recognition tasks, e.g., (Charniak and Goldman, 1993; Horvitz and Paek, 1999; Zukerman, 2001). Charniak and Goldman's system handled complex narratives, using a BN and marker passing for plan recognition. It automatically built and incrementally extended a BN from propositions read

[*] This research was supported in part by Australian Research Council grant A49927212.



(a) Original argument (Arg)
The neighbour reported a *heated* argument between Mr Green and Mr Body last week
AND Mr Green was *seen* in the garden at 11
-> Mr Body was murdered by Mr Green *[likely]*
ML(Arg | IG_Arg)
(b) Top-ranked IG_Arg
G was in garden at 11
N reported argument
G murdered B
ML(IG_Arg |IG_SysInt)
(c) IG_SysInt for best SysInt
N reported argument
N heard argument
G argued with B
G and B were enemies
G was in garden at 11
G was in garden at time of death
G had opportunity
G had motive
G murdered B
ML(IG_SysInt |SysInt) = 0
(d) SysInt (Top Candidate)
Interpret Sentences
Construct an Argument Interpretation
MML Evaluation of the Interpretation

Figure 1: Interpretation and MML evaluation

in a story, so that the BN represented hypotheses that became plausible as the story unfolded. In contrast, we use a BN to constrain our understanding of the propositions in an argument, and apply the MML principle to select a plausible interpretation. Both Horvitz and Paek's system and Zukerman's handled short dialogue contributions. Horvitz and Paek used BNs at different levels of an abstraction hierarchy to infer a user's goal in information-seeking interactions with a Bayesian Receptionist. Zukerman used a domain model and user model represented as a BN, together with linguistic and attentional information to infer a user's goal from a short-form rejoinder. However, the combination of these knowledge sources was based on heuristics.

The MML principle is a model selection technique which applies information-theoretic criteria to trade data fit against model complexity. Selected applications which use MML are listed in `http://www.csse.monash.edu.au/~dld/Snob.application.papers`.

3 Argument Interpretation Using MML

According to the MML criterion, we imagine sending to a receiver the shortest possible message that describes an NL argument. When a good interpretation is found, a message which encodes the NL argument in terms of this interpretation will be shorter than the message which transmits the words of the argument directly.

A message that encodes an NL argument in terms of an interpretation is composed of two parts: (1) instructions for building the interpretation, and (2) instructions for rebuilding the original argument from this interpretation. These two parts balance the need for a concise interpretation (Part 1) with the need for an interpretation that matches closely the original argument (Part 2). For instance, the message for a concise interpretation that does not match well the original argument will have a short first part but a long second part. In contrast, a more complex interpretation which better matches the original argument may yield a shorter message overall. As a result, in finding the interpretation that yields the shortest message for an NL argument, we will have produced a plausible interpretation, which hopefully is the intended interpretation. To find this interpretation, we compare the message length of the candidate interpretations. These candidates are obtained as described in Section 4.

3.1 MML Encoding

The MML criterion is derived from Bayes Theorem: $\Pr(D\&H) = \Pr(H) \times \Pr(D|H)$, where D is the data and H is a hypothesis which explains the data.

An optimal code for an event E with probability $\Pr(E)$ has message length $\mathrm{ML}(E) = -\log_2 \Pr(E)$ (measured in bits). Hence, the message length for the data and a hypothesis is:

$$\mathrm{ML}(D\&H) = \mathrm{ML}(H) + \mathrm{ML}(D|H)\,.$$

The hypothesis for which $\mathrm{ML}(D\&H)$ is minimal is considered the best hypothesis.

Now, in our context, *Arg* contains the argument, and *SysInt* an interpretation generated by our system. Thus, we are looking for the *SysInt* which yields the shortest message length for

$\mathrm{ML}(\mathit{Arg}\&\mathit{SysInt}) = \mathrm{ML}(\mathit{SysInt}) + \mathrm{ML}(\mathit{Arg}|\mathit{SysInt})$

The first part of the message describes the interpretation, and the second part describes how to reconstruct the argument from the interpretation. To calculate the second part, we rely on an intermediate representation called *Implication Graph* (*IG*). An Implication Graph is a graphical representation of an argument, which represents a basic "understanding" of the argument. It is composed of simple implications of the form $Antecedent_1\,Antecedent_2\ldots Antecedent_n \Rightarrow Consequent$ (where $\Rightarrow$ indicates that the antecedents imply the consequent, without distinguishing between causal and evidential implications). IG_{Arg} represents an understanding of the input argument. It contains propositions from the underlying representation, but retains the structure of the argument. IG_{SysInt} represents an understanding of a candidate interpretation. It is directly obtained from *SysInt*. Hence, both its structure and its propositions correspond to the underlying representation. Since both IG_{Arg} and IG_{SysInt} use domain propositions and have the same type of representation, they can be compared with relative ease.

Figure 1 illustrates the interpretation of a small argument, and the calculation of the message length of the interpretation. The interpretation process obtains IG_{Arg} from the input, and *SysInt* from IG_{Arg} (left-hand side of Figure 1). If a sentence in *Arg* matches more than one domain proposition, the system generates more than one IG_{Arg} from *Arg* (Section 4.1). Each IG_{Arg} may in turn yield more than one *SysInt*. This happens when the underlying representation has several ways of connecting between the nodes in IG_{Arg} (Section 4.2). The message length calculation goes from *SysInt* to *Arg* through the intermediate representations IG_{SysInt} and IG_{Arg} (right-hand side of Figure 1). This calculation takes advantage of the fact that there can be only one IG_{Arg} for each *Arg–SysInt* combination. Hence,

$$\begin{aligned}\Pr(Arg\,\&\,SysInt) &= \Pr(Arg, IG_{Arg}, SysInt)\\ &= \Pr(Arg|IG_{Arg}, SysInt)\Pr(IG_{Arg}|SysInt)\Pr(SysInt)\\ &\overset{\text{cond. ind.}}{=} \Pr(Arg|IG_{Arg})\Pr(IG_{Arg}|SysInt)\Pr(SysInt)\end{aligned}$$

Thus, the length of the message required to transmit the original argument from an interpretation is

$$\text{ML}(Arg\,\&\,SysInt) = \tag{1}$$
$$\text{ML}(Arg|IG_{Arg}) + \text{ML}(IG_{Arg}|SysInt) + \text{ML}(SysInt)$$

That is, for each candidate interpretation, we calculate the *length* of the message which conveys:

- *SysInt* – the interpretation,
- $IG_{Arg}|SysInt$ – how to obtain the belief and structure of IG_{Arg} from *SysInt*,[1] and
- $Arg|IG_{Arg}$ – how to obtain the sentences in *Arg* from the corresponding nodes in IG_{Arg}.

The interpretation which yields *the shortest message* is selected (the message-length equations for each component are summarized in Table 1).

3.2 Calculating ML(SysInt)

In order to transmit *SysInt*, we simply send its propositions and the relations between them. A standard MML assumption is that the sender and receiver share domain knowledge. Hence, one way to send *SysInt* consists of transmitting how *SysInt* is extracted from the domain representation. This involves selecting its propositions from those in the domain, and then choosing which of the possible relations between these propositions are included in the interpretation. In the case of a BN, the propositions are represented as nodes, and the relations between propositions as arcs. Thus the message length for *SysInt* in the context of a BN is

$$\log_2 C^{\#\_\text{nodes}(domainBN)}_{\#\_\text{nodes}(SysInt)} + \log_2 C^{\#\_\text{incident\_arcs}(SysInt)}_{\#\_\text{arcs}(SysInt)} \tag{2}$$

3.3 Calculating ML(IG$_{\textbf{Arg}}$|SysInt)

The message which describes IG_{Arg} in terms of *SysInt* (or rather in terms of IG_{SysInt}) conveys how IG_{Arg} differs from the system's interpretation in two respects: (1) belief, and (2) argument structure.

3.3.1 Belief differences

For each proposition N in both IG_{SysInt} and IG_{Arg}, we transmit any discrepancy between the belief stated in the argument and the system's belief in this proposition (propositions that appear in only one IG are handled by the message component which describes structural differences). The length of the message required to convey this information is

$$\sum_{N \in IG_{Arg} \cap IG_{SysInt}} \text{ML}(Bel(N, IG_{Arg})|Bel(N, IG_{SysInt}))$$

where $Bel(N, IG_x)$ is the belief in proposition N in IG_x. Assuming an optimal message encoding, we obtain

$$\sum_{N \in IG_{Arg} \cap IG_{SysInt}} -\log_2 \Pr(Bel(N, IG_{Arg})|Bel(N, IG_{SysInt})) \tag{3}$$

which expresses discrepancies in belief as a probability that the argument will posit a particular belief in a proposition, given the belief held by the system in this proposition. We have modeled this probability using a function which yields a maximum proba-

[1]We use IG_{SysInt} for this calculation, rather than *SysInt*. This does not affect the message length because the receiver can obtain IG_{SysInt} directly from *SysInt*.

bility mass when the belief in proposition N according to the argument agrees with the system's belief. This probability gradually falls as the discrepancy between the belief stated in the argument and the system's belief increases, which in turn yields an increased message length.

3.3.2 Structural differences

The message which transmits the structural discrepancies between IG_{SysInt} and IG_{Arg} describes the structural operations required to transform IG_{SysInt} into IG_{Arg}. These operations are: node insertions and deletions, and arc insertions and deletions. A node is inserted in IG_{SysInt} when the system cannot reconcile a proposition in the given argument with any proposition in its domain representation. In this case, the system proposes a special *Escape* (wild card) node. Note that the system does not presume to understand this proposition, but still hopes to achieve some understanding of the argument as a whole. Similarly, an arc is inserted when the argument mentions a relationship which does not appear in IG_{SysInt}. An arc (node) is deleted when the corresponding relation (proposition) appears in IG_{SysInt}, but is omitted from IG_{Arg}. When a node is deleted, all the arcs incident upon it are rerouted to connect its antecedents directly to its consequent. This operation, which models a small inferential leap, preserves the structure of the implication around the deleted node. If the arcs so rerouted are inconsistent with IG_{Arg} they will be deleted separately.

For each of these operations, the message announces how many times the operation was performed (e.g., how many nodes were deleted) and then provides sufficient information to enable the message receiver to identify the targets of the operation (e.g., which nodes were deleted). Thus, the length of the message which describes the structural operations required to transform IG_{SysInt} into IG_{Arg} comprises the following components:

$$\begin{aligned}&\mathrm{ML}(\text{node insertions}) + \mathrm{ML}(\text{node deletions}) + \\ &\mathrm{ML}(\text{arc insertions}) + \mathrm{ML}(\text{arc deletions}) \qquad (4)\end{aligned}$$

- **Node insertions =** number of inserted nodes plus the penalty for each insertion. Since a node is inserted when no proposition in the domain matches a statement in the argument, we use an insertion penalty equal to T_M – the probability-like score of the worst acceptable word-match between a statement and a proposition (Section 4.1). Thus the message length for node insertions is
$$\log_2(\#\_\text{nodes\_ins}) + \#\_\text{nodes\_ins} \times (-\log_2 T_M) \quad (5)$$
- **Node deletions =** number of deleted nodes plus their designations. To designate the nodes to be deleted, we select them from the nodes in *SysInt* (or IG_{SysInt}):
$$\log_2(\#\_\text{nodes\_del}) + \log_2 C^{\#\_\text{nodes}(IG_{SysInt})}_{\#\_\text{nodes\_del}} \quad (6)$$
- **Arc insertions =** number of inserted arcs plus their designations plus the direction of each arc. (This component also describes the arcs incident upon newly inserted nodes.) To designate an arc, we need to select a pair of nodes (head and tail) from the nodes in IG_{SysInt} and the newly inserted nodes. However, some nodes in IG_{SysInt} are already connected by arcs. These arcs must be subtracted from the total number of arcs that can be inserted, yielding
$$\begin{aligned}\#\_\text{poss\_arc\_ins} \;=\; & C_2^{\#\_\text{nodes}(IG_{SysInt}) + \#\_\text{nodes\_ins}} \\ & - \#\_\text{arcs}(IG_{SysInt})\end{aligned}$$
We also need to send 1 extra bit per inserted arc to convey its direction. Hence, the length of the message that conveys arc insertions is:
$$\log_2(\#\_\text{arcs\_ins}) + \log_2 C^{\#\_\text{poss\_arc\_ins}}_{\#\_\text{arcs\_ins}} + \#\_\text{arcs\_ins} \quad (7)$$
- **Arc deletions =** number of deleted arcs plus their designations.
$$\log_2(\#\_\text{arcs\_del}) + \log_2 C^{\#\_\text{arcs}(IG_{SysInt})}_{\#\_\text{arcs\_del}} \quad (8)$$

3.4 Calculating ML(Arg|IG$_{\text{Arg}}$)

The given argument is structurally equivalent to IG_{Arg}. Hence, in order to transmit *Arg* in terms of IG_{Arg} we only need to transmit how each statement in *Arg* differs from the canonical statement generated for the matching node in IG_{Arg} (Section 4.1). The length of the message which conveys this information is
$$\sum_{N \in IG_{Arg}} \mathrm{ML}(\text{Sentence}_N \text{ in } Arg | N)$$
where Sentence$_N$ in *Arg* is the sentence in the original argument which matches the proposition for node N in IG_{Arg}. Assuming an optimal message encoding, we obtain
$$\sum_{N \in IG_{Arg}} -\log_2 \Pr(\text{Sentence}_N \text{ in } Arg | N) \quad (9)$$

We approximate $\Pr(\text{Sentence}_N \text{ in } Arg | N)$ using the score returned by the comparison function described in Section 4.1.

Table 1: Summary of Message Length Calculation

ML(*Arg* & *SysInt*)	Equation 1
ML(*SysInt*)	Equation 2
ML(IG_{Arg}|*SysInt*)	
belief operations	Equation 3
structural operations	Equations 4, 5, 6, 7, 8
ML(*Arg*|IG_{Arg})	Equation 9

4 Proposing Interpretations

Our system generates candidate interpretations for an argument by first postulating propositions that match the sentences in the argument, and then finding different ways to connect these propositions – each variant is a candidate interpretation.

4.1 Postulating propositions

We currently use a naive approach for postulating propositions. For each sentence S_{Arg} in the given argument we generate candidate propositions as follows. For each proposition N in the domain, the system proposes a canonical sentence CS_N (produced by a simple English generator). This sentence is compared to S_{Arg}, yielding a match-score for the pair (S_{Arg}, N). When a match-score is above a threshold T_M, we have found a candidate interpretation for S_{Arg}. For example, the proposition [G was in garden at 11] in Figure 1(b) is a plausible interpretation of the input sentence "Mr Green was seen in the garden at 11" in Figure 1(a). Some sentences may have no propositions with match-scores above T_M. This does not automatically invalidate the argument, as it may still be possible to interpret the argument as a whole, even if a few sentences are not understood (Section 3.3).

The match-score for a sentence S_{Arg} and a proposition N – a number in the [0,1] range – is calculated using a function which compares words in S_{Arg} with words in CS_N. The goodness of a word-match depends on the following factors: (1) level of synonymy – the number of synonyms the words have in common (according to WordNet, Miller *et al.*, 1990); (2) position in sentence; and (3) part-of-speech (PoS) – obtained using MINIPAR (Lin, 1998). That is, a word W_{i,CS_N} in position i in CS_N matches perfectly a word $W_{j,S_{Arg}}$ in position j in sentence S_{Arg}, if both words are exactly the same, they are in the same sentence position, and they have the same PoS. The match-score between W_{i,CS_N} and $W_{j,S_{Arg}}$ is reduced as their level of synonymy falls, and as the difference in their sentence position increases. The match-score of two words is further reduced if they have different PoS. In addition, the PoS affects the penalty for a mismatch, so that mismatched non-content words are penalized less than mismatched content words.

The match-scores between a sentence and its candidate propositions are normalized, and the result used to approximate Pr(S_{Arg}|N), which is required for the MML evaluation (Section 3.4).[2]

4.2 Connecting the propositions

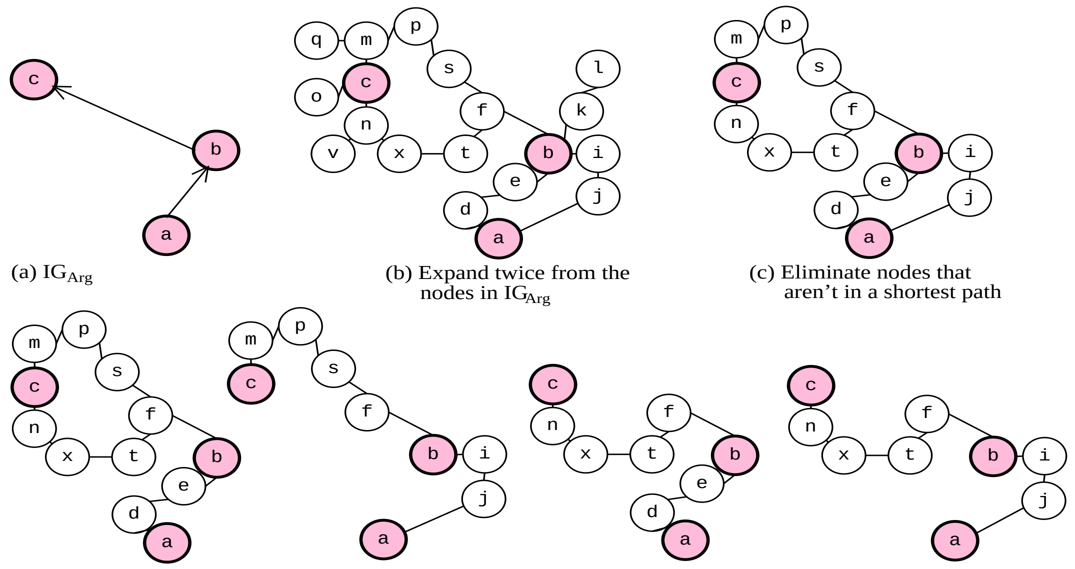
Since more than one node may match each of the sentences in an argument, there may be more than one IG_{Arg} that is consistent with the argument. For instance, the sentence "Mr Green was seen in the garden at 11" in Figure 1(a) matches both [G was in garden at 11] and [N saw G in garden] (although the former has a higher probability). If the other sentences in Figure 1(a) match only one proposition, two IGs that match the argument will be generated – one for each of the above alternatives.

Figure 2 illustrates the remainder of the interpretation-generation process with respect to one IG_{Arg}. This process consists of finding connections between the nodes in IG_{Arg}; eliminating superfluous nodes; and generating sub-graphs of the resulting graph, such that all the nodes in IG_{Arg} are connected (Figures 2(b), 2(c) and 2(d), respectively). The connections between the nodes in IG_{Arg} are found by applying two rounds of inferences from these nodes (spreading outward). These two rounds enable the system to "make sense" of an argument with small inferential leaps (Zukerman, 2001). If upon completion of this process, some nodes in IG_{Arg} are still unconnected, the system rejects IG_{Arg}. This process is currently implemented in the context of a BN. However, any representation that supports the generation of a connected argument involving a given set of propositions would be appropriate.

5 Evaluation

Our evaluation consisted of an automated experiment where the system interpreted noisy versions of its own arguments. These arguments were generated from different sub-nets of its domain BN, and they were distorted at the BN level and at the NL level. At the BN level, we changed the beliefs in the nodes, and we inserted and deleted nodes and arcs. At the NL level, we distorted the wording of the propositions in the resultant arguments. All

[2] We are implementing a more principled model for sentence comparison which yields more accurate probabilities.



(a) $\mathrm{IG}_{\mathrm{Arg}}$

(b) Expand twice from the nodes in $\mathrm{IG}_{\mathrm{Arg}}$

(c) Eliminate nodes that aren't in a shortest path

(d) Candidates are all the subgraphs of (c) that connect the nodes in $\mathrm{IG}_{\mathrm{Arg}}$ (4 of the 9 candidates are shown)

Figure 2: Argument interpretation process

these distortions were performed for BNs of different sizes (3, 5, 7 and 9 arcs). Our measure of performance is the edit-distance between the original BN used to generate an argument, and the BN produced as the interpretation of this argument. For instance, two BNs that differ by one arc have an edit-distance of 2 (one addition and one deletion), while a perfect match has an edit-distance of 0.

Overall, our results were as follows. Our system produced an interpretation in 86% of the 5400 trials. In 75% of the 5400 cases, the generated interpretations had an edit-distance of 3 or less from the original BN, and in 50% of the cases, the interpretations matched perfectly the original BN. Figure 3 depicts the frequency of edit distances for the different BN sizes under all noise conditions. We plotted edit-distances of 0, . . ., 9 and > 9, plus the category NI, which stands for "No Interpretation". As shown in Figure 3, the 0 edit-distance has the highest frequency, and performance deteriorates as BN size increases. Still, for BNs of 7 arcs or less, the vast majority of the interpretations have an edit distance of 3 or less. Only for BNs of 9 arcs the number of NIs exceeds the number of perfect matches.

We also tested each kind of noise separately, maintaining the other kinds of noise at 0%. All the distortions were between 0 and 40%. We performed 1560 trials for word noise, arc noise and node insertions, and 2040 trials for belief noise, which warranted additional observations. Figures 4,

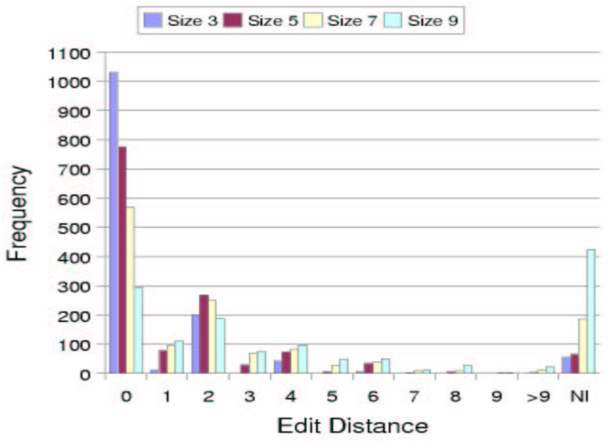


Figure 3: Frequency of edit-distances for all noise conditions (5400 trials)

5 and 6 show the recognition accuracy of our system (in terms of average edit distance) as a function of arc, belief and word noise percentages, respectively. The performance for the different BN sizes (in arcs) is also shown. Our system's performance for node insertions is similar to that obtained for belief noise (the graph was not included owing to space limitations). Our results show that the two main factors that affect recognition performance are BN size and word noise, while the average edit distance remains stable for belief and arc noise, as well as for node insertions (the only exception occurs for 40% arc noise and size 9 BNs). Specifically, for arc noise, belief noise and node insertions, the average

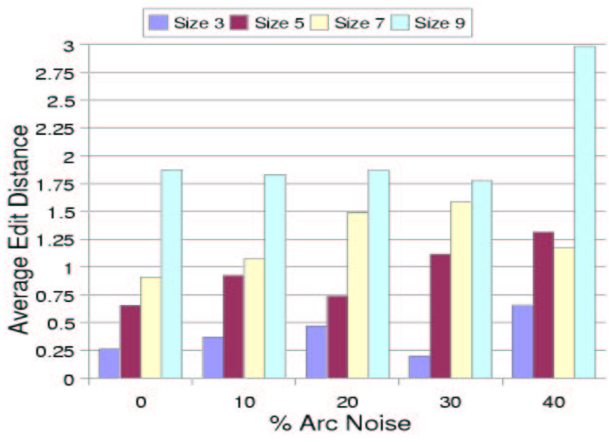


Figure 4: Effect of arc noise on performance (1560 trials)

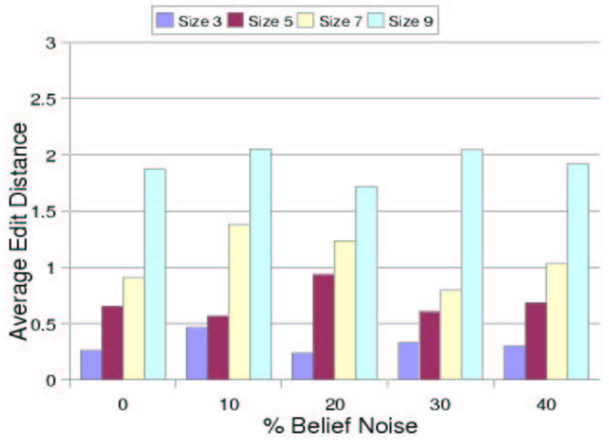


Figure 5: Effect of belief noise on performance (2040 trials)

edit distance was 3 or less for all noise percentages, while for word noise, the average edit distance was higher for several word-noise and BN-size combinations. Further, performance deteriorated as the percentage of word noise increased.

The impact of word noise on performance reinforces our intention to implement a more principled sentence comparison procedure (Section 4.1), with the expectation that it will improve this aspect of our system's performance.

6 Conclusion

We have offered a mechanism which produces interpretations of segmented NL arguments. Our application of the MML principle enables our system to handle noisy conditions in terms of wording, beliefs and argument structure, and allows us to isolate the effect of the underlying knowledge representation on the interpretation process. The results of our automated evaluation were encouraging, with interpretations that match perfectly or almost-perfectly the source-BN being generated in 75% of the cases under all noise conditions.

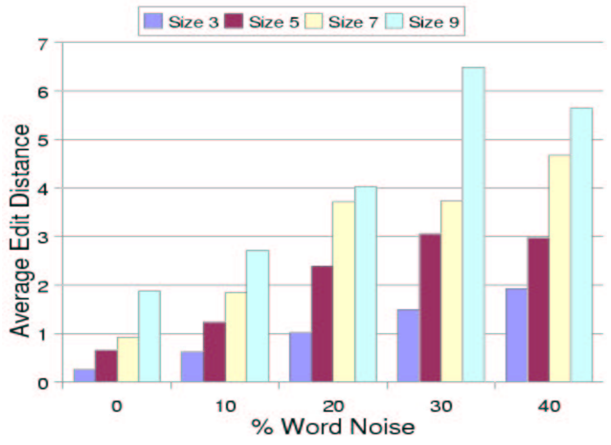


Figure 6: Effect of word noise on performance (1560 trials)

References

Eugene Charniak and Robert P. Goldman. 1993. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):50–56.

Eric Horvitz and Tim Paek. 1999. A computational architecture for conversation. In *UM99 – Proceedings of the Seventh International Conference on User Modeling*, pages 201–210, Banff, Canada.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, San Mateo, California.

C.S. Wallace and D.M. Boulton. 1968. An information measure for classification. *The Computer Journal*, 11:185–194.

Ingrid Zukerman. 2001. An integrated approach for generating arguments and rebuttals and understanding rejoinders. In *UM01 – Proceedings of the Eighth International Conference on User Modeling*, pages 84–94, Sonthofen, Germany.