Using Edit Distance to Analyse Errors in a Natural Language to Logic Translation Corpus

Dave Barker-Plummer
Center for the Study of Language and Information
Stanford University
Stanford, CA 94305, USA
dbp@stanford.edu

Robert Dale
Center for Language Technology
Macquarie University
Sydney, NSW 2109, Australia
Robert.Dale@mq.edu.au

Richard Cox
Faculty of Information Technology
Monash University
Clayton, VIC 3168, Australia
rcox@inf.ed.ac.uk

ABSTRACT

We have assembled a large corpus of student submissions to an automatic grading system, where the subject matter involves the translation of natural language sentences into propositional logic. Of the 2.3 million translation instances in the corpus, 286,000 (approximately 12%) are categorized as being in error. We want to understand the nature of the errors that students make, so that we can develop tools and supporting infrastructure that help students with the problems that these errors represent.

With this aim in mind, this paper describes an analysis of a significant proportion of the data, using edit distance between incorrect answers and their corresponding correct solutions, and the associated edit sequences, as a means of organising the data and detecting categories of errors. We demonstrate that a large proportion of errors can be accounted for by means of a small number of relatively simple error types, and that the method draws attention to interesting phenomena in the data set.

1. INTRODUCTION

As reported in [2], we have developed a large corpus of student submissions to an automatic grading system, where the subject matter involves the translation of natural language (NL) sentences into first-order logic (FOL). The translation exercises are a subset of the exercises in *Language, Proof and Logic* (LPL; [6]), a courseware package consisting of a textbook together with desktop applications which students use to complete exercises.[1] The translation exercises contain a total of 275 distinct translatable sentences; an abridged example of such an exercise is shown in Figure 1.[2] We refer to each translation of a sentence submitted by a student as a **translation instance**; the corpus contains 4.5 million translation instances, collected from more than 55,000 individual students over the period 2001–2010. In this paper, we focus on that subset of the data concerned with translations into propositional logic; this accounts for 2,340,306 translation instances. The Grade Grinder, our automatic grading system, categorizes 286,106 of these (approximately 12%) of these instances as being in error; Figure 2 shows some common incorrect solutions for the first sentence in the exercise shown in Figure 1.

Currently, our grading system simply indicates whether or not the student has obtained a correct answer. Our interest is in (a) developing tools and supporting infrastructure that can help students with these errors by providing better feedback; and (b) using the evidence from the errors that students make to drive broader pedagogical improvements. In order to achieve these goals, we first need to acquire an understanding of the kinds of errors that students make.

Of course, any teacher of logic has intuitions about what aspects of logic students find hard and why. Any good logic textbook embodies a distilling of what is essentially folk wisdom based on teacher experience. In contrast, our goal is to use our large dataset of errors as part of an evidence-based approach to diagnosing, assisting and correcting a wide range of errors. There have been data-driven studies in the past (see, for example, our own [3]), but these have been based on relatively small amounts of data, and so there is a risk that they are essentially anecdotal.

Our goal, then, is to see what we can learn by looking at larger and more comprehensive sets of student behaviour in this domain. In this paper, we look at how **edit distance** can be used as a tool for characterising the nature of the errors in our dataset. We demonstrate that a large proportion of errors can be accounted for by means of relatively simple error types, and discuss how some of these errors correspond to specific kinds of problems that students struggle with.

The structure of the remainder of this paper is as follows. In Section 2, we briefly review related work, describe the data we work with, and outline our general approach to its analysis. In Section 3, we provide some summary statistics on the subset of the data that it is the focus of the present paper. We then move on in Sections 4 and 5 to a detailed analysis of the errors in this data set, and make some observations based on our analyses. Finally, in Section 6, we draw some conclusions and outline our future plans.

2. BACKGROUND

2.1 Related Work

Students have particular difficulty with the formalisation tasks that underpin logic, and so it is not surprising that there is a body of work that seeks to understand the nature of these difficulties, often with the aim of improving on-line tutoring systems; for example, Barnes et al. [5] studied students' logic learning trajectories in order to extend Deep

[1] See `http://lpl.stanford.edu`.
[2] The 'Δ' column is explained later.

➶ **Exercise 7.12** (Translation) Translate the following English sentences into FOL. Your translations will use all of the propositional connectives.

(1) *If* a *is a tetrahedron then it is in front of* d.
(2) a *is to the left of or right of* d *only if it's a cube.*
(3) c *is between either* a *and* e *or* a *and* d.
...
(19) a *is large just in case* d *is small.*
(20) a *is large just in case* e *is.*

Figure 1: An example exercise (7.12) from LPL

Ex.	N	Correct	Incorrect	Δ
1	429	$\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, d)$	$\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, b)$	1
2	254	$\neg\mathsf{FrontOf}(a, d) \rightarrow \neg\mathsf{Tet}(a)$	$\mathsf{FrontOf}(a, d) \rightarrow \mathsf{Tet}(a)$	2
3	160	$\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, d)$	$\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(d)$	1
4	96	$\mathsf{Tet}(a) \rightarrow \mathsf{FrontOf}(a, d)$	$\mathsf{Tet}(a) \leftrightarrow \mathsf{FrontOf}(a, d)$	1

Figure 2: Correct and incorrect solutions to Exercise 7.12.1

Thought, a logic proof tutor, by adding a hint generator.

The work carried out by various groups has helped to identify specific categories of error that recur in formalisation tasks.

Logical misunderstandings are cognitive misconceptions, such as when a student substitutes the biconditional (*if and only if*) for material implication (*if ... then*), or when they reverse the antecedent and consequent of an implication, writing $A \rightarrow B$ when the correct answer is $B \rightarrow A$. These misunderstandings are often related to what we might think of as **natural language transfer errors**, where some property true of the natural language sentence, typically to do with its structure or ordering of elements, is also assumed to be true of the formal translation. An early study by Clement, Lochhead and Monk [8] investigated such translation difficulties in mathematics: students were instructed to write an equation representing the statement *There are six times as many students as professors at this university*, and were told to use S for the number of students and P for the number of professors. The most common error consisted of reversing the variables in the equation, writing 6S = P. An explanation of this phenomenon offered by Clement et al. was **word-order matching**, in which the student orders terms in their equation in a way that matches the order of keywords in the problem statement.

A more general source of difficulty for students lies in the stance they must adopt towards formal language, which, compared to everyday discourse, is more concise, less redundant, and has high information density and precision. Otherwise everyday terms are often used in formal contexts with quite specific meanings; Pimm [13] provides numerous examples of such 'borrowed terms' in mathematics (e.g., *face*, *mean*, *real*, *natural*). Consequently, a student may bring expectations in regard to the use of particular terms that are not in fact borne out.

Of course, errors which are not unique to formalisation also manifest themselves in formalisation tasks. **Slips** [14] are such a type of error; in the context of the translation of natural language into logic, an example would be an attentional lapse resulting in a student using the wrong letter to denote a constant (e.g., b instead of d).

Our own work to-date has focussed upon characterizing logic translation tasks in terms of the grammatical and other features of the English sentences to be translated, and upon the elucidation of the nature of errors made by students on those tasks. In previous work we have carried out a detailed analysed of the errors that students make when performing translation tasks using data from just one of the LPL exercises [3], and we have compared the errors that students make when translating into logic as compared to diagrammatic form [9], together with the effects of visual and spatial content [4]. We have also looked at students' individual learning trajectories by examining their repeated submissions to the Grade Grinder [1].

2.2 The Data

LPL contains 33 exercises that involve translations of natural language into logic. Each exercise involves multiple sentences to be translated, so these 33 exercises consist of 275 distinct natural language sentences that students are asked to translate. A complete list of the sentences to be translated, together with the number of translation instances we have for each natural language sentence, and the proportion of instances that were deemed by the Grade Grinder to be in error, is provided in [2], along with other detailed statistics.

When a student submits an exercise, the Grade Grinder assigns a **status** to each sentence in that exercise. An exercise is considered correct if every sentence in the submission is assigned a status of **correct**, which indicates that the sentence is provably equivalent to a gold-standard correct answer. Other common statuses include **incorrect**, which indicates that the sentence is not provably correct, and **ill-formed**, which indicates that the student's submission is not a well-formed formula of first-order logic.

Each entry in our corpus records the identity of the sentence being translated, a unique identifier for the student, the particular string submitted by the student, and the status that it was assigned by the Grade Grinder. Auxilliary information concerning the time at which the submission was made, and other similar meta-information, is also recorded, but plays no role in the current study.

LPL introduces the language of first-order logic in two stages. In the first, the quantifier-free fragment (propositional logic) is introduced, while consideration of the full first-order language is deferred until the second stage. All of the data for this study is drawn from exercises in propositional logic, i.e., from Chapters 1–8 of the textbook; this results in a subcorpus of 286,106 translation instances.

2.3 Approach

In our earlier explorations of the data [1; 3; 4; 9], we used a somewhat labor-intensive approach to uncovering the nature of the errors that students make: this involved writing exercise-specific regular expressions to pick out relevant properties of each submitted answer. Such an approach is appropriate for initial forays into large datasets, since it gives a sense of the range and variety of the data; but it doesn't scale up to data collections like the one focussed on

here. We need a way to manage the dataset that provides some structure, and provides some means for determining how subsequent more-detailed analyses might be prioritised.

Given our pedagogical aims, a sensible strategy is to focus on what is common and repeated in the data; if there are some mistakes that many students make, it obviously makes sense to develop techniques to address these issues, rather than focussing on much rarer errors whose probability of reoccurrence is small. But this immediately presents us with a problem. The natural language sentences that students are asked to translate are deliberately and necessarily quite varied, often combining a variety of phenomena both in their surface forms and in their corresponding logical translations. This is entirely appropriate if we want to test a student's ability to deal with non-trivial formalisation problems that are not unidimensional; but this property of the data also makes it difficult to identify and characterise the important features that are shared across instances of error. Put simply: any given incorrect solution provided by a student may simultaneously represent a number of different errors, and the same error or misunderstanding may manifest itself in different sentences in different ways.

Our approach is to seek a method of characterising, for any pairing of a correct answer and an incorrect translation provided by a student, the delta between the two. This delta captures the essence of the particular error or set of errors that the student made; we can then look for similarities across the deltas found in different contexts, with the aim of identifying common problems that recur. Essential to making this work is the availability of a representation that abstracts out just the right characteristics of the delta.

An approach to characterising error that has been used in other domains—most notably in the context of natural language spelling correction—is that of **edit distance** [12]. The idea here is much-studied and well-known: the difference between two objects can be characterised by a sequence of edit operations (typically, the insertion, deletion and transposition of elements), and the length of this sequence is the edit distance between the two objects.[3] In the context of spelling correction, the objects are strings whose constituent elements are characters; the edit operations are therefore the insertion, deletion and transposition of characters. The misspelling *teh* is then correctable to *the* by a single transposition, giving an edit distance of 1. An oft-cited finding is that 80% of spelling errors are edit distance 1 from their corresponding correct forms [10], supporting a view that most errors are simple; in the interface to a spelling correction program, this provides a basis for ordering potential corrections in terms of increasing edit distance.

Technically, the approach just described uses the notion of **string edit distance**. However, our sense is that the tree structure of well-formed logic sentences will play a role in characterising the nature of the errors that students make; we already saw hints of this in our earlier work [3], where we identified antecedent–consequent reversal (where, effectively, two subtrees in a formula are transposed) as a common error type. Consequently, rather than string edit distance, we use **tree edit distance**. Here, the edit distance is defined between ordered labelled trees, i.e., trees

[3]Substitution is also sometimes considered an atomic operation, but this can be achieved by a deletion followed by an insertion.

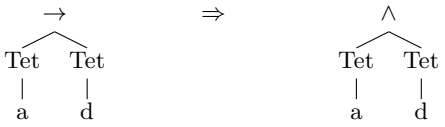

Figure 3: Relabelling

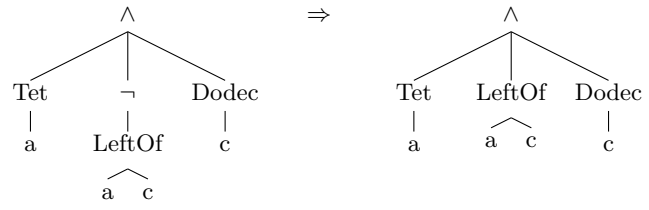

Figure 4: Node deletion/insertion

in which each node has an associated label, and the order of the children of a node is considered significant. We will use the term **tree** as a shorthand for ordered, labelled trees. The edit distance between two trees is defined as the minimum number of edit operations that must be applied to one tree, the **source**, to make it identical to another, the **target**. Critical to this definition is a set of edit operations that we have available to make this transformation. In our formulation, which is typical, three edit operations are available: **relabeling**, in which the label on the node of a tree is changed; **deletion**, in which a node is deleted from the tree, with any children of the deleted node becoming children of the deleted node's parent (in order); and **insertion**, the inverse of deletion. Figure 3 shows a relabelling operation applied to a tree; Figure 4 shows a case where a node has been deleted in the move from left-to-right, or equivalently inserted from right-to-left.

The edit distance problem in trees has been studied extensively (see [7] for a survey); our implementation is of the algorithm due to Klein [11]. More efficient algorithms exist, but the additional complexity of these algorithms is not justified in our context primarily because our trees are quite small.

It is important to note that translation exercises do not have unique solutions. Student submissions are graded as correct if they are provably equivalent to a gold-standard correct answer known to the system. Consequently, the corpus contains a set of submitted correct answers in addition to the set of submitted incorrect answers. This presents a problem for our approach: which of the possible correct answers should we compute the distance from when considering a given incorrect answer? Our response to this problem is to choose the correct answer for which the edit distance is minimized. This strategy essentially embodies the assumption that simple errors are more likely.

Figure 2 shows some example pairs for translations of Sentence 7.12.1 in Figure 1. Example 1 demonstrates an error with edit distance 1, as indicated by the column headed 'Δ'. Here, the student has used the constant `b` when `a` was required by the correct answer. Note that a different correct answer is used in Example 2: a simple description of this error is that the student has reversed the order of the antecedent and consequent, but an alternative explanation

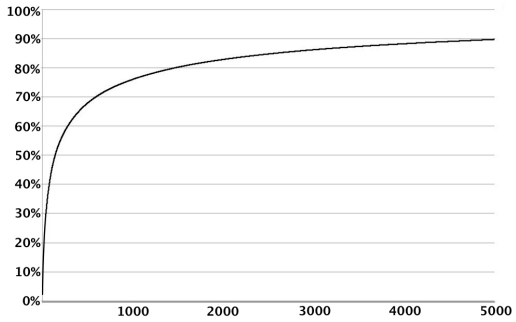

Figure 5: Cumulative coverage by error type

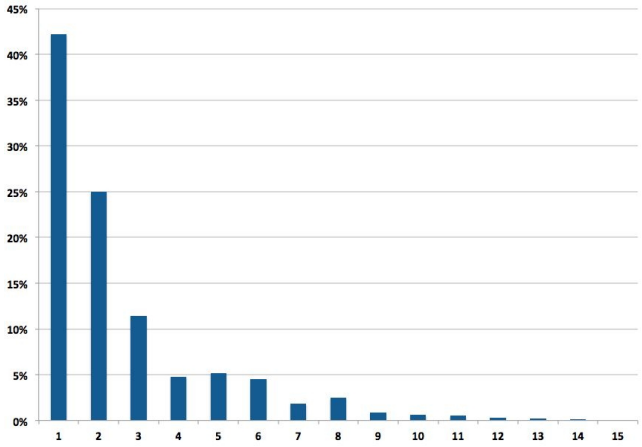

Figure 6: Proportions of data by edit distance

is the one shown here, which is that they have omitted negation symbols from the contrapositive of the answer in Example 1. This transformation has an edit distance of 2, and is 'cheaper' than the corresponding reversal.

3. ERROR DISTRIBUTION

An **error instance** is an occasion of a particular student submitting a particular incorrect translation for a given NL sentence; an **error type** is a pairing of an incorrect translation with the correct translation of that sentence, abstracted away from its particular occurrences in the data. The point here is that the data manifests varying numbers of instances for each distinct error type: some incorrect translations are very common (i.e., many students provide the same incorrect translation) and some are rare (i.e., only one or two students produce that particular error).

Our 286,106 error instances are distributed across a total of 27,151 error types, but the distribution is very skewed: the 19 most frequent error types account for 20% of the data, and 142 types account for 50% of the data. Figure 5 shows the cumulative coverage of the error types in order of their frequency in the data.

A note is in order here in regard to our use of absolute counts of errors, rather than proportions of incorrect solutions for each translation. It might be thought that the latter would provide a more realistic measure of where the difficulties lie for students: if only 10 students attempt one question and nine get it wrong, this suggests that the question is more difficult than one where 1000 students attempt the question and 500 get it wrong, despite the larger absolute count of errors in the second case. However, the fact remains that some sentences are translated more often than others; and if we want to maximise the impact of our work, it is the questions which generate high numbers of errors, irrespective of the number of attempts, that are most important. Consequently, in this paper we report statistics in terms of absolute counts, although for other purposes proportions will be more informative.

We note that the most frequent error types in the data are edit distance 1 from their respective correct translations. We call such errors **unit errors** or **units**. Unit errors account for 42.24% of the incorrect answer instances, and errors with edit distance 2 account for a further 24.97% of the incorrect answer instances, for a total of 67.22%. Figure 6 breaks down the error instances by edit distance. There is a long tail, and so the graph in this figure is truncated: the largest edit distance represented in the corpus is 73 (with one instance).

Given the preponderance of relatively simple errors, in the remainder of this paper we focus on the simple and easily detectable edit sequences as a means of organising and analysing the data. This is not to say that the more complex errors are not important; they still account for a significant proportion of the data. However, the analyses required to identify the relevant patterns are correpondingly more complex. Our strategy is therefore to explore the simpler errors first, with the possibility of subsequently factoring these elements out from the more complex errors to further assist in managing the complexity.

4. HOMOGENEOUS EDITS

Our goal is to use edit distance as a way of organising the errored data in the corpus. We are interested in two things here, both of which are explored further below.

1. We want to see whether edit distance as a metric provides a useful way to organise and analyse the data; for example, we might expect that it would allow us to organise the data in terms of the complexity of the errors contained.

2. In terms of exploring the particular errors that students make, it is conceivable that the edit sequences required to transform an error into its corrected form might reveal something about the nature of the errors made.

As a first step, we organise relabellings into three subcategories: RELABEL$_{Conn}$, RELABEL$_{Const}$, and RELABEL$_{Pred}$. The basis for this categorization is the syntactic category of the tokens that participate in the relabeling. The tree edit distance algorithm is domain-independent, and thus it has no information about the roles of the symbols at the nodes of the trees. At the logical level, we distinguish between **constant symbols**, which serve to name objects, **predicate symbols**, which serve to name properties and relations, and **logical connectives**, which connect together sentences into larger sentences. When a node relabeling operation involves symbols which are both of the same syntactic category, then

Ex.	Correct	Incorrect	Instances	Edit Sequence
1	$\neg$Between$(c,a,b) \wedge$ $\neg$(FrontOf$(c,a) \vee$ FrontOf(c,b))	$\neg$Between$(c,a,b) \vee$ $\neg$(FrontOf$(c,a) \wedge$ FrontOf(c,b))	521	[RELABEL$_{Conn}$ $\vee$ at $\langle\rangle$ to $\wedge$ at $\langle\rangle$, RELABEL$_{Conn}$ $\wedge$ at $\langle.2.1\rangle$ to $\vee$ at $\langle.2.1\rangle$]
2	(LeftOf$(a,d) \vee$ RightOf$(a,d)) \rightarrow$ Cube(a)	(LeftOf$(a,d) \vee$ RightOf$(a,d)) \rightarrow$ Cube(d)	899	[RELABEL$_{Const}$ d at $\langle.2.1\rangle$ to a at $\langle.2.1\rangle$]
3	SameCol$(b,e) \wedge$ SameRow$(e,d) \wedge$ SameCol(d,a)	SameCol$(b,e) \wedge$ SameCol$(e,d) \wedge$ SameCol(d,a)	769	[RELABEL$_{Pred}$ SameCol at $\langle.2\rangle$ to SameRow at $\langle.2\rangle$]
4	Cube(a)	Cub(a)	1	[RELABEL$_{Pred}$ Cub at $\langle\rangle$ to Cube at $\langle\rangle$]
5	$\neg$Cube$(b) \rightarrow$ Tet(c)	Cube$(b) \rightarrow \neg$Tet(c)	1262	[INSERT $\neg$ at $\langle.1\rangle$, DELETE $\neg$ at $\langle.2\rangle$]

Table 1: Some representative error types

we categorize this relabeling as the appropriate subcategory above. A relabeling which involves labels from more than one syntactic category—for example, a constant being replaced by a connective—is called RELABEL$_{Mixed}$. The RELABEL$_{Mixed}$ edit is very rare in the corpus; in fact there are only three homogeneous RELABEL$_{Mixed}$ edit sequences, each of which accounts for one error instance. We note that a similar subtyping of the INSERT and DELETE is possible, but we have not yet made use of this granularity.

Example 1 in Table 1 presents an instance of RELABEL$_{Conn}$, where the student wrote $\vee$ when $\wedge$ was expected; Example 2 in that table presents an instance of RELABEL$_{Const}$, where the student wrote d where a was expected; and Example 3 in the table presents an example of RELABEL$_{Pred}$, where the student wrote SameCol in place of SameRow.

Having identified these different classes of edit operations, we can distinguish two kinds of edit sequences: those that are **homogeneous** and those that are **heterogeneous**. We call an edit sequence **homogeneous** if all of the edit operations in the sequence are of the same type: for example, they might be all INSERT or all RELABEL$_{Conn}$ operations. Any edit sequence that is not homogeneous is **heterogeneous**.

Homogeneous edit sequences are significant from our point of view because they have the potential to be identified and analysed more easily. For example, a homogeneous edit sequence that contains only RELABEL$_{Const}$ edits indicates that the student has made a mistake concerning only the constants and their positions within the answer sentence, but the other features of the sentence are correct. The analysis of these errors based on the information in the edit sequences that they produce is likely to be more tractable, while the analysis of heterogeneous edit sequences, by contrast, will be more involved.

Moreover, homogeneous edit sequences account for more than 60% of the error instances in the corpus. Focussing on these edit sequences enables us to make significant headway in understanding the nature of the errors that students make without having to interpret convoluted edit sequences.

Within the homogeneous edits, we distinguish two categories: **relabellings**, and **insertions and deletions**. A homogeneous relabeling edit consists only of relabelling operations of a single type (for example, all RELABEL$_{Const}$ or all RELABEL$_{Conn}$), indicating that the structure of the formula tree that the student produced was identical to that which was expected (as in Figure 3 shown earlier). Homogeneous insertion and deletion sequences, on the other hand, consist of only INSERT or DELETE operations (although with possibly different operands), meaning that the structure of the formula tree produced by the student was different from that which was expected (Figure 4 shows a case where a single DELETE has occurred).

4.1 Relabellings

4.1.1 RELABEL$_{Conn}$

The largest category of homogeneous relabellings in the data are sequences of one or more RELABEL$_{Conn}$ edits. These account for 25.10% of the error instances, with 23.45% of the total data set being units; this indicates that students often get the overall structure of the solution correct, but are confused about which logical connectives to use to capture the intended meaning. A simple example of a pair of sentences exhibiting this pattern is the student sentence Tet(a) $\wedge$ Tet(b) being provided when the correct answer is Tet(a) $\rightarrow$ Tet(b); as shown in Figure 3, here the connective relabeling occurs at the root of the tree.

It is interesting to look more closely at the particular relabellings that students use. Table 2 presents a confusion matrix that shows, for each pair of connectives, how often one connective is found when the other is expected. The numbers here indicate the proportion of attempts that generated the error in each case; so, for example, the pair $\langle$Found: $\leftrightarrow$, Expected: $\rightarrow\rangle$ has a value of 10.33%, indicating that in 10.33% of the situations where a biconditional was expected, the student instead used an implication.[4] Example 4 in Figure 2 provides an example of this extremely common error type.

The other notable confusion here is the use of conjunction when disjunction is required; this occurs in 2.74% of possible cases. For both these confusions, the inverse substitution is also quite prominent, but in neither case to the same extent. This data provides insights into the particular connectives that cause problems for students, and suggest where in the data more detailed analysis might be performed to build an understanding of student misconceptions.

[4]Note that these numbers are for homogeneous relabellings only, i.e., cases where the structure of the formula provided is otherwise correct. It is highly likely that this same error occurs as part of more complex heterogeneous edit sequences, but these are harder to isolate.

found ⇓	expected ∧	∨	→	↔
∧	–	2.74%	0.50%	0.44%
∨	0.79%	–	0.07%	0.06%
→	0.45%	0.09%	–	10.33%
↔	0.20%	0.09%	2.54%	–

Table 2: Confusion of connectives

found ⇓	expected a	b	c	d	e	f
a	-	0.11%	0.20%	0.07%	0.10%	0.31%
b	0.05%	-	0.15%	0.47%	0.48%	0.04%
c	0.05%	0.13%	-	0.12%	0.42%	0.02%
d	0.20%	0.08%	0.09%	-	0.13%	0.05%
e	0.07%	0.05%	0.10%	0.08%	-	0.08%
f	0.02%	0.01%	0.01%	0.04%	0.02%	-

Table 3: Confusion of constants

4.1.2 RELABEL$_{Const}$

Our results show that 13.20% of the error instances are accounted for by homogenous RELABEL$_{Const}$ edits, of which the first 3.32% are units. A confusion matrix illustrating students' constant substitutions is provided in Table 3.

In [3], we reported data from a smaller-scale study of constant substitutions based on data from just one of the LPL exercises. We noted that this kind of error seemed to interact with (1) the use of the constant a in a sentence; (2) whether a was the first-mentioned constant in the sentence; and (3) whether the constant names were alphabetically adjacent (e.g. ⟨a, b, c⟩) as opposed to being alphabetically 'gappy' (e.g. ⟨b, e, d⟩). What we referred to as the **gappiness effect** was statistically significant, being magnified when (1) the letters used as constants were not alphabetically adjacent; and (2) the first constant name mentioned in the sentence was a. Table 3 shows that the most common constant substitution is ⟨Found: b, Expected: e⟩. The NL sentence with which this substitution is most frequently associated is *Neither* **e** *nor* **a** *is to the right of* **c** *and to the left of* **b**. The next two most common substitution patterns are ⟨Found: c, Expected: e⟩ and ⟨Found: b, Expected: d⟩. These three frequent substitutions are ones in which the substituted constant appears earlier in the alphabetic sequence than the expected constant; this is consistent with the operation of a gappiness effect.

Table 3 shows that the fourth largest proportion is associated with ⟨Found: a, Expected: f⟩. An inspection of the NL sentences suggests that this is associated with cases where they contain a and another constant, b. This particular substitution is rarer when a is present as the only other constant, or when the other constants are not a and b (e.g., c and d). This apparent trend warrants further investigation. The fifth most frequent substitution is ⟨Found: d, Expected: a⟩. One sentence (**a** *is to the left or right of* **d** *only if it's a cube*) stands out in relation to this error. Here there is anaphoric reference to an object that is a cube; the intention is that the cube should be labelled a, but clearly many students resolve *it* to the cube with identity d.

4.1.3 RELABEL$_{Pred}$

The RELABEL$_{Pred}$ operation appears relatively infrequently in the data corpus. Only 4.51% of the error instances are accounted for by homogenous RELABEL$_{Pred}$ edits, of which the first 3.32% are units.

An interesting example of this is where the correct sentence is $\mathsf{SameCol}(\mathsf{b},\mathsf{e}) \wedge \mathsf{SameRow}(\mathsf{e},\mathsf{d}) \wedge \mathsf{SameCol}(\mathsf{d},\mathsf{a})$ while the incorrect answer is $\mathsf{SameCol}(\mathsf{b},\mathsf{e}) \wedge \mathsf{SameCol}(\mathsf{e},\mathsf{d}) \wedge \mathsf{SameCol}(\mathsf{d},\mathsf{a})$. This is the second most frequent homogeneous RELABEL$_{Pred}$ sequence, accounting for 769 error instances. Like gappiness, this appears to be the result of students' inability to adopt a sufficiently careful stance toward the formalization task.

In this case, one predicate from the expected language (here, the Blocks World language) is replaced by another from the same language;[5] however, we also find cases where a predicate is replaced by one from outside the language. In the most common instance of this type, instead of the expected answer $2:00 < 2:05$ we find the answer $\mathsf{Earlier}(2:00, 2:05)$: The student has here used a predicate derived from the lexical content of the NL sentence, but this predicate does not exist in the expected formal language (the Pets World language). This error type accounts for 1976 error instances.[6]

4.2 Insertions and Deletions

So far we have discussed only relabeling operations, but the tree edit distance computation also allows for the inclusion of INSERT and DELETE operations in the edit sequences. Each such edit operation inserts or deletes a node into the source tree as it is transformed to the target. Unlike relabeling operations, INSERT and DELETE modify the *structure* of the trees in order to align them. Just as a homogeneous relabeling edit sequence may relabel nodes at various points in the tree, a homogeneous sequence of DELETE operations could delete many nodes from the tree. However, we might expect that deletions would come in groups. For example, consider again the error in Example 2 of Figure 2, but this time computing the distance from the correct answer shown for Example 1 in that table. A simple description of this pair is that the incorrect answer has the antecedent and consequent of the implication switched. The shortest edit sequence that achieves this is one which: (1) deletes the node labelled Tet from its position in the consequent (one operation); (2) inserts it as the new antecedent (one operation); (3) removes the node labelled a from its original position (one operation); and (4) inserts a as a child of Tet in its new location, for a total edit distance of 4. But notice that here the INSERT and DELETE operations come in pairs, and that the deleted nodes form a subtree of the original tree. We call these **local** insert or delete sequences. We believe that these local sequences may be of particular interest, because they appear to aggregate as higher level operations.

There is an interesting asymmetry in our data. There are many more homogeneous edit sequences involving INSERT (15.51% of all error instances, with the first 4.34% being units) than there are using DELETE (only 2.16% of all error instances, with the first 1.10% being units). This indicates that students are more likely to provide translations

[5] Most exercises in LPL use the language of a Blocks World; there are also exercises that make use of a 'Pets World' of pets and their owners.

[6] Note that the notion of being outside the language also includes the possibility that a student commits a simple typing error, such as typing Cub where Cube is expected; see Example 4 in Table 1.

that omit information present in the NL sentence than they are to add information. This seems plausible, since any additional information would have to come from somewhere, and this would have to be from outside the source sentence. It is much more likely that students misread the sentence, failing to notice or express relevant information that is present.

One example of students providing 'extra' information in their answer is in an exercise containing the sentence *Notice that f (the large dodecahedron in the back) is not in front of a. Use your first sentence to say this.* In this exercise, students are looking at a picture of a situation in which the sentences are true. We intend the student to translate *f is not in front of a*, and provide a parenthetical hint to help them identify f in the picture. 687 students additionally provide the information that f is a large dodecahedron. This finding suggests that some students are so focussed on the operations required by the translation task that their awareness of problem context is diminished, a process akin to the phenomenon of **suspension of sense-making** proposed in [15].[7]

5. HETEROGENEOUS EDITS

As noted above, homogeneous edit sequences account for just over 60% of the data; consequently, 40% of the error instances are heteregeneous, i.e. they consist of more than one type of edit operation.

Our intuition is that particular subsequences (or, more likely, particular subsets) of the edits that make up these edit sequences will correspond to characteristic errors. Identifying these patterns would then allow us to detect the same or similar errors occurring in quite different contexts. However, we observe that, as with so many pattern recognition problems, getting the representation right is crucial, and it is quite likely that the particular set of edit operations we have chosen to use is not optimal for this purpose.

As a case in point: in our earlier work, we identified antecedent–consequent reversal as a particularly prevalent error. From the perspective of tree manipulation, this corresponds to a swapping of two subtrees. Such an operation is only indirectly perceptible in terms of our current edit operations. Although we could derive a characterisation of this error in terms of the lower-level operations, an alternative would be to 'reify' it in terms of higher-level tree manipulation operations; in particular, we might make tree insertion and deletion be primitive operations in determining edit sequences and the corresponding edit distances. The technical challenge here is to retain the tractability of the edit distance computation if these more complex operations are permitted.

As an exploration of the potential of such higher-level edit operations, we observe that one characteristic of the relocation of a subtree is that it will consist of n DELETE operations paired with n INSERT operations.[8] It turns out that 4.94% of the error instances in our data set consist of n INSERTs and n DELETEs, which we might think of as the simplest form of heterogeneous edits. Note that this category does not include longer edit sequences that contain equal numbers of INSERTs and DELETEs in conjunction with other edits, so the actual number of such pairs is likely to be larger.[9]

DELETE	INSERT	Count
¬	¬	8581
folly	folly	1406
a	a	410

Figure 7: The three most common symbol movements

One must be careful in interpreting such numbers. It is certainly not the case, for example, that all of these correspond to subtree movements: apart from the fact that other constraints must be met for a subtree movement to be in evidence, the reality is that the vast majority of these paired INSERTs and DELETEs consist of exactly one instance of each, and so they correspond to movements of single symbols rather than larger structures: such movements account for 4.58% of the error instances in the corpus. But even this simpler observation is an important finding: Figure 7 shows the three most common symbol movements that occur in the data set. This reveals that by far the most common error of this type is misplaced negation, as in Example 5 in Table 1; this is clearly a place where students need assistance. The other two cases shown in this table are most likely to be instances of argument movement (although it is conceivable that there are other causes of the phenomena), which arises whenever a student provides an answer that has the arguments in the wrong order.

6. CONCLUSIONS

Students find formalisation hard; but the ability to formalise problems underpins many fields, and so any means we can find to improve student abilities in this regard is important and valuable. Our ultimate goal is to develop evidence-based pedagogy in this area, and so we want to take advantage of our large data set of student translations to guide this activity.

Unfortunately, there are no established analytical tools for exploring this kind of data. In this paper, we have presented the results of an initial exploration that uses the notions of edit distance and edit operation sequence as a means to better understanding what it is that students do wrong when they make mistakes in NL to FOL translation. We have been encouraged by the outcomes:

1. Unit errors and edit distance 2 errors account for approx 67% of our data, permitting tractable use of tree edit distance as a useful data mining method in our domain.

2. Results from the application of tree edit distance validate our earlier, pilot findings from analyses of a subsample of the data analysed in this paper. In particular, the analysis provides support for our earlier

[7]As a result of finding this phenomenon in the data, we have subsequently configured the Grade Grinder to accept this as a correct answer.

[8]There are, of course, a number of other constraints, most notably the relationship between the addresses of the nodes in each pair of INSERTs and DELETEs.

[9]We say only likely because it is correspondingly more difficult to determine whether the INSERTs and DELETEs are related to each other; in more complex sequences, it is also possible that some INSERTs or DELETEs belong to some other higher-level transformation.

observations in regard to gappiness in constant substitutions, and antecedent–consequent reversal [3]. Validating previous results by triangulation using different methods promotes confidence in the findings.

As noted throughout this paper, a number of more specific findings provide important stepping-off points for developing a better understanding of the nature of common student errors.

Of course, much remains to be done. So far, we have shown that the techniques described here allow us to identify particular errors that manifest themselves in many different circumstances; but this is really a *descriptive* analysis of the data, and does not necessarily correspond to the kinds of *explanations* for these errors that we might need if we want to provide appropriate assistance to students. In some cases the explanation may seem to follow straightforwardly from the description, but in other cases we need to dig deeper into the context (what are the characteristics of the NL sentence being translated? what is the student's history of error?) in order to provide an explanation. Moving in this direction is a major priority in our future work.

Orthogonal to this line of development, we should note also that we have focussed here on what we call homogenous edits, these being the simpler to identify and analyse. As noted earlier, these account for 60% of the data, so they allow us to make considerable headway; but this leaves the 40% of the data that correspond to heterogeneous edits still to be explored. One strategy we aim to pursue here is to see whether we can decompose heterogeneous edit sequences into partitions, some of which may correspond to the homogenous edit sequences we have already explored; this would then leave a simpler residue still to be characterised. However, we suspect that there may be more mileage to be gained by revising the set of edit operations used to be closer to the kinds of operations that are more conceptually valid in the domain; for example, allowing insertion and deletion of entire subtrees as atomic operations may provide a better way of characterising some of the mistakes that students make.

Armed with these kinds of tools, we hope to be able to develop a better understanding of the mistakes that students make, and to provide appropriate assistance.

7. ADDITIONAL AUTHORS

Additional authors: Alex Romanczuk, CSLI/Stanford University, email: `aproman@stanford.edu`.

8. REFERENCES

[1] D. Barker-Plummer, R. Cox, and R. Dale. Dimensions of difficulty in translating natural language into first order logic. In *Second International Conference on Educational Data Mining*, pages 220–229. Cordoba Spain, 2009.

[2] D. Barker-Plummer, R. Cox, and R. Dale. Student Translations of Natural Language into Logic: The Grade Grinder Corpus Release 1.0. In *Proceedings of The 4th International Conference on Educational Data Mining*, pages 51–60, 6th–8th July 2011.

[3] D. Barker-Plummer, R. Cox, R. Dale, and J. Etchemendy. An empirical study of errors in translating natural language into logic. In V. Sloutsky, B. Love, and K. McRae, editors, *Proceedings of the 30th Annual Cognitive Science Society Conference*, pages 505–510. Lawrence Erlbaum Associates, 2008.

[4] D. Barker-Plummer, R. Dale, and R. Cox. Impedance effects of visual and spatial content upon language-to-logic translation accuracy. In C. Hoelscher, T. F. Shipley, and L. Carlson, editors, *Proceedings of the 32nd Annual Cognitive Science Society Conference*, pages 3259–3264. Lawrence Erlbaum Associates, 2011.

[5] T. Barnes, J. Stamper, L. Lehmann, and M. Croy. A pilot study on logic proof tutoring using hints generated from historical student data. In T. B. R. Baker and J. Beck, editors, *Proceedings of the 1st International Conference on Educational Data Mining (EDM 2008)*, pages 197–201. International Educational Data Mining Society, 2008.

[6] J. Barwise, J. Etchemendy, G. Allwein, D. Barker-Plummer, and A. Liu. *Language, Proof and Logic*. CSLI Publications and University of Chicago Press, September 1999.

[7] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217–239, 2003.

[8] J. Clement, J. Lochhead, and G. Monk. Translation difficulties in learning mathematics. *The American Mathematical Monthly*, 88(4):286–290, 1981.

[9] R. Cox, R. Dale, J. Etchemendy, and D. Barker-Plummer. Graphical revelations: Comparing students' translation errors in graphics and logic. In G. Stapleton, J. Howse, and J. Lee, editors, *Proceedings of the Fifth International Conference on the Theory and Application of Diagrams*. Lecture Notes in Computer Science LNAI 5223, Berlin: Springer Verlag, 2008.

[10] F. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[11] P. N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms (ESA)*, pages 91–102. Springer-Verlag, 1998.

[12] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[13] D. Pimm. *Speaking Mathematically*. Routledge and Kegan Paul, 1987.

[14] J. Reason. *Human Error*. Cambridge University Press, 1990.

[15] L. Verschaffel, B. Greer, and E. de Corte. *Making Sense of Word Problems*. Swets and Zeitlinger, 2000.