Triangulating product and process data: quantifying alignment units with keystroke data

Michael Carl

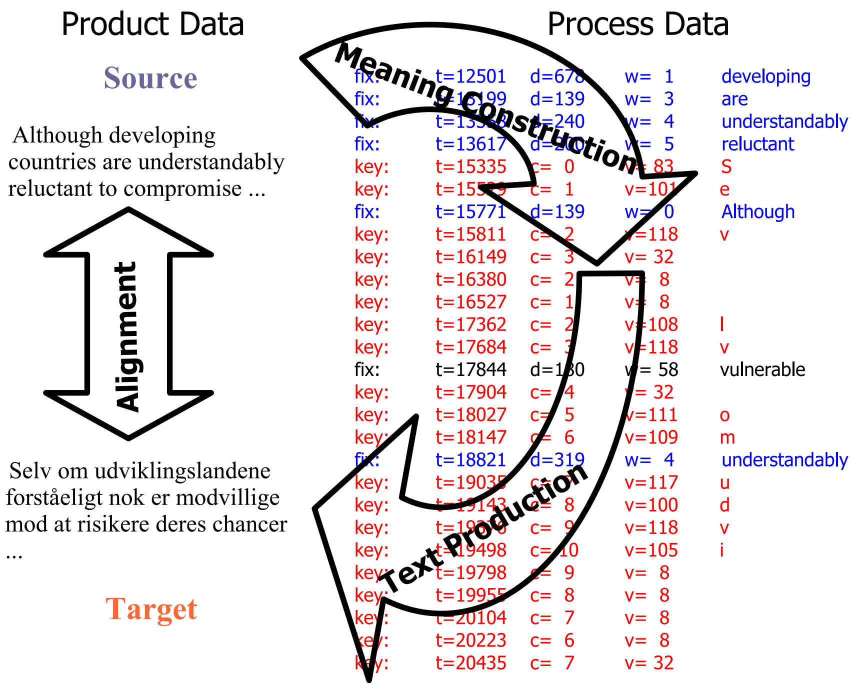
[B]y testing hypotheses based on qualitative data against quantitative data, and vice versa, I believe we can soon begin to make stronger and more informed guesses about translation. (Jakobsen 1999, 19)

Abstract

The paper discusses a method to triangulate process and product data. We suggest converting Translog data into a relational format which contains both process and product data. We outline how this representation allows us to retrieve and correlate the various dimensions of the data more easily. The concept of Alignment Unit (AU) is introduced and contrasted with that of Translation Unit (TU). While AUs refer to translation equivalences in the source and target texts of the product data, TUs refer to cognitive entities that can be observed in the process data. With an (almost) exhaustive fragmentation of the source and target texts into AUs, we are able to distribute and allocate the entire set of keystroke data to appropriate AUs. Using the properties of the keystroke data, AUs are quantified in a novel way which enables us to visualise and investigate the structure of translation production on a fine-grained scale.

1 Introduction

Triangulation is the application of combined research methodologies, theories, or data sources to double (or triple) check scientific results: “If observations of comparable phenomena remain stable and convergent from the perspective of different methods, the possibility that they are mere artefacts are reduced.” (Jakobsen 1999, 18ff) By combining multiple observations, theories, methods, and empirical evidence, one can hope to overcome the biases of a single method.


Product Data
Source
Although developing countries are understandably reluctant to compromise ...
Alignment
Selv om udviklingslandene forståeligt nok er modvillige mod at risikere deres chancer ...
Target
Process Data
Meaning Construction
Text Production

```
fix:   t=12501   d=678    w=  1    developing
fix:   t=13199   d=139    w=  3    are
fix:   t=13358   d=240    w=  4    understandably
fix:   t=13617   d=200    w=  5    reluctant
key:   t=15335   c=  0    v= 83    S
key:   t=15529   c=  1    v=101    e
fix:   t=15771   d=139    w=  0    Although
key:   t=15811   c=  2    v=118    v
key:   t=16149   c=  3    v= 32
key:   t=16380   c=  2    v=  8
key:   t=16527   c=  1    v=  8
key:   t=17362   c=  2    v=108    l
key:   t=17684   c=  3    v=118    v
fix:   t=17844   d=180    w= 58    vulnerable
key:   t=17904   c=  4    v= 32
key:   t=18027   c=  5    v=111    o
key:   t=18147   c=  6    v=109    m
fix:   t=18821   d=319    w=  4    understandably
key:   t=19035   c=  7    v=117    u
key:   t=19143   c=  8    v=100    d
key:   t=19276   c=  9    v=118    v
key:   t=19498   c= 10    v=105    i
key:   t=19798   c=  9    v=  8
key:   t=19955   c=  8    v=  8
key:   t=20104   c=  7    v=  8
key:   t=20223   c=  6    v=  8
key:   t=20435   c=  7    v= 32
```

Figure 1. The Figure represents the dimensions of the User Activity Data (UAD): on the left the product data and on the right side the process data. Both types of data provide different aspects of the translation session.

Jakobsen (2006) suggests *methodological triangulation* of Translog keyboard data to be used and verified against retrospective interviews, TAPs, video/audio recordings, screen recording, and eye tracking in order to achieve "more informed guesses about translation"; he also mentions a "within method triangulation" in which, for instance "quantitative time-delay data [would be checked] against typing speed data, etc." (Jakobsen 1999, 19)

Since methodological triangulation requires additional hardware and a large synchronisation overhead and introduces technical problems which occur when combining heterogeneous media, we here suggest a 'within method' for Translog process and product data, which is also referred to as *data triangulation*, and show some of the potentials it has of detecting and quantifying dependencies.

"*Translog* is a computer program that logs keyboard activity involved in making a translation" (Jakobsen 1999, 1) and its output contains both the product and the process data that emerges during a translation session. In previous publications (Carl et al. 2008; Carl and Jakobsen 2009) we have used the term *User Activity Data* (UAD) to subsume any kind of data which is consulted or generated by a translator during a translation session. Figure 1 represents the relation between process and product data as it is currently produced by Translog. The left side shows the static part of the UAD, the source text (ST) and the target text (TT), together with their correspondence links. The right side plots the dynamic process data which consists of keyboard actions and gaze behaviour. According to Jakobsen (2006, 104):

> The keystroke data can be interpreted quantitatively and unambiguously in terms of either text production, text elimination or cursor navigation [...] actions. [...] This means that the keystroke record is also immediately interpretable in terms of linguistic units (letters, words, clauses, etc.) giving the researcher full access to studying the typing process by which a text is produced.

While this statement is certainly true, the notion of "word" or "clause" in the second part of the implication is unsatisfactory in a translation context, for instance when two ST words are translated into one, or a coherent ST segment is transformed into a discontinuous sequence in the TT. The monolingual entities do not immediately reveal the cross-linguistic dependencies which we want to study.

Translog itself only provides "a simple calculation of the total number of keystrokes and the number of text production keystrokes" (Jakobsen 2006, 104). This is certainly not sufficient to study translation processes in detail, but it is stated that "utility software" can be created that automatically segments a logfile into suitable units. Below such a utility software is suggested which segments the product data into "Alignment Units" (AUs) and triangulates those units with keystroke process data.

An alignment indicates "which parts of the source text correspond with which parts of the target text" (Dagan et al. 1993, 1). A Translation Unit (TU), in contrast, reflects, for some, entities of cognitive processes or the "translator's focus of attention at a given time in the translation process" (Alves and Vale 2009, 254). This definition implies that TUs are dynamic units that change over time, so that the same ST word may successively be part of different TUs. Thus, the term 'TU' seems to refer to sequences of activities which may be observed in the *process data*, while we will use AUs to indicate translation correspondences in the static *product data*.

While we do not exactly know what TUs actually are,[1] the problem is less serious for AUs. It is possible to align single words and continuous or discontinuous sequences of m-to-n words even across sentences boundaries. AUs are usually not below word level, AUs are likely to be shorter than TUs,

[1]See Alves and Vale (2009) and Dragsted (2004, 14ff) for excellent overviews of different usages of the term 'TU'.

and AUs can be directly observed, discussed and in most cases agreed upon between different annotators. But most importantly, AUs do not change in time and do not have the dynamic aspect that TUs have.

A large number of trainable alignment programs exist (e.g. Dagan et al. 1993, Och and Ney 2000, Kromann 2003, to name a few) which can be used to (semi-) automatically align segments of a ST with their correspondences in a TT. An isomorphism between AUs and TUs is often assumed in a machine translation context, where aligned texts serve as a basis for the extraction of TUs to be subsequently reused in the machine translation system.

It is, however, unclear whether and to what extent such an isomorphism also exists in the case of human translation. One attempt to define TUs in human translation process data has been undertaken by Dragsted (2004). With the intention to detect TUs on the basis of long pauses between keystrokes, Dragsted (2004, 142) states that "[i]ntuitively, the TU must contain both SL and TL elements". She comes to a number of conclusions about the length (in words) of TUs under different conditions. However, a differentiation between TUs and AUs might be instrumental in dealing with inconsistent segmentation in the product and process data. For instance, the Danish-English AU: "er det planen $\leftrightarrow$ we expect" shows a long translation pause in the process data between "we" and "expect", which indicates incompatible segmentation boundaries in both types of data. A distinction into separate TUs on the process side and AUs on the product side would allow us to correlate and "triangulate" these units without the need to give up or change definitions.

A systematic triangulation of TUs and AUs is beyond the scope of this study. Rather, we shall prepare the ground for such an investigation and correlate and quantify AUs as detected in the product data with keystroke data. In order to do so in a systematic and automatic manner, we need to:

1. add further alignment information to define the AUs. We use the DTAG software package[2] which allows us to semi-automatically align the ST and TT;
2. transform the Translog product and process data into a representation which better reflects the phenomena which we want to investigate;
3. use a query language to retrieve all keystroke activities for each AU.

[2]DTAG `http://code.google.com/p/copenhagen-dependency-treebank/` is a project aimed at creating linguistically annotated text collections (treebanks) on the basis of the dependency-based grammar formalism Discontinuous Grammar. A number of mono- and bilingual semi-automatic annotation and visualisation tools can be downloaded from the web site.

```
<W cur="0" top="76" btm="110" lft="1" rgt="107">Although</W>
<W cur="9" top="76" btm="110" lft="115" rgt="243">developing</W>
<W cur="20" top="76" btm="110" lft="251" rgt="359">countries</W>
<W cur="30" top="76" btm="110" lft="367" rgt="405">are</W>
<W cur="34" top="76" btm="110" lft="413" rgt="595">understandably</W>
```

Figure 2. The ST representation includes word information (i.e. the surface form of the word), information on word location on the screen (i.e. the top-left and bottom-right pixel positions of the box containing the word) and the word position in the text (i.e. the cursor position of the word's first letter).

```
<W cur="0" top="511" btm="544" lft="21" rgt="69">Selv</W>
<W cur="5" top="511" btm="544" lft="77" rgt="115">om</W>
<W cur="8" top="511" btm="544" lft="123" rgt="332">udviklingslandene</W>
<W cur="26" top="511" btm="544" lft="340" rgt="459">forståeligt</W>
<W cur="38" top="511" btm="544" lft="467" rgt="510">nok</W>
<W cur="42" top="511" btm="544" lft="518" rgt="542">er</W>
```

Figure 3. The TT representation is structurally identical to the ST representation, containing information about words and their location.

An exhaustive fragmentation of the product data into AUs would then also provide a framework for the study of the process data, since each keystroke may be considered to contribute to one (or more) AU(s). The number and distribution of keystrokes which fall into each AU constitute a novel way of classification and a yet unstudied field of research.

In Section 2, we describe how we transform the Translog product data into a format that makes it better suitable for the triangulation we are aiming at. Section 3 describes how we transform the Translog keystroke data into a format for better interrogation. The aim is to represent UAD in a kind of relational data structure which allows us to automatically detect and correlate the various dimensions of the product and process data. Section 4 outlines a query language that allows us to relate the different dimensions of the data in a formalised and exhaustive manner. Our particular aim is to correlate and quantify AUs with keystroke data. Section 5 describes a special operator of the query language which makes it possible to retrieve all keyboard activities which belong to a given TT string. With these concepts we are ready to discuss the triangulation of AUs and keystroke data in Section 6.

2 Product data

As outlined in Figure 1, the Translog output contains product and process data, which is coded in XML. For our purposes, we have to transform the

UAD into a representation that can be more easily correlated. The product data consists of the ST and the TT, which can be extracted from the Translog data.[3] The representation we are aiming at is shown in Figures 2 and 3. In addition we need alignment information to know which pieces in the SL text correspond to which pieces of the TL text. We represent alignments as shown in Figure 4.

```
<A src="0" tgt="0"/>
<A src="0" tgt="5"/>
<A src="9" tgt="8"/>
<A src="20" tgt="8"/>
<A src="30" tgt="42"/>
<A src="34" tgt="26"/>
```

Figure 4. Alignment information consists of m-to-n word correspondences between the ST and the TT. The above representation shows four AUs which connect the ST in Figure 2 and the TT in Figure 3. 1: “Although $\leftrightarrow$ Selv om”, 2: “developing countries $\leftrightarrow$ udviklingslan-dene” 3: “are $\leftrightarrow$ er”, and 4: “understandably $\leftrightarrow$ forståeligt”. Note that “nok” in the Danish text segment in Figure 3 is not aligned to any English ST word.

We have based our data triangulation experiments on four English-Danish translation sessions and used the DTAG toolkit (Kromann 2003) to semi-automatically align the translations of the four subjects. Table 1 summarises properties of the translations and the number of aligned AUs. While the English ST has 100 words[4] and was identical in all cases, the length of the translations varies between 101 and 112 words. Not all words in the product data could be aligned (see Section 5 for more details) and in addition in some cases several words were subsumed in one AU so that also the number of AUs is different for each translation.

	A2	K	M	S
Number of words in TT	102	101	112	101
Aligned words in TT	97	92	106	98
Aligned words in ST	100	97	98	96
Number of AUs	82	71	80	69

Table 1. Summary of properties of product data: number of words in the translations, number of aligned words and number of AUs. The source text has 100 words, see Appendix.

[3]The word location information in Figures 2 and 3 is only available in the Translog ‘premium’ version (`http://www.translog.dk/`), which is also able to collect gaze data. In the ‘ordinary’ version, only the cursor information is available which has to be extracted post-hoc from the log file.

[4]The ST is reproduced in the Appendix

The translations were produced by professional translators and the texts were manually aligned by three translation students with no particular instruction as to how alignment should be done. The alignments were not analysed in detail. The difference in number of AUs reflects their average number of words, and may indicate a different perception of compositionality. Each translation alignment took approximately 30 minutes and was in most cases straightforward.

3 Keystroke data

As outlined in Jakobsen (1999, 2006) Translog logs all keyboard and mouse activities. Different keyboard and/or mouse actions can be used for the insertion and deletion of text, and for cursor navigation. For instance, deletions can be achieved through backspace, or by pressing the delete key. A piece of text can also be deleted by first highlighting it and then hitting the deletion or backspace key or even by overwriting the highlighted sequence with other characters. Highlighting of the text can, again, be achieved by various means, e.g. by using the mouse button or by a combination of using shift and left, right, up, or down keys, or even pressing shift control and left, right, up, or down keys, etc. All these events are coded differently in Translog output. The full information is needed to replay the translation session in a natural way.

For an analysis of how a translation emerges, a more reduced representation is sufficient. We are likely to be not so much interested in *how* the cursor was moved to a particular position in the text, but rather *that* it was there at a particular point in time. We may also be not so much interested in knowing whether a sequence was deleted by first highlighting it using the mouse or a combination of keystrokes, or whether it was deleted using repeated delete or backspace keys. Rather, for our intended analysis it is sufficient to observe that the sequence in question was deleted. For knowing what happens to the text, we are basically only interested in knowing the insertions and the deletions that take place at any point in time.

Figures 5, 6, and 7 show examples of the simplified key logging information: each line represents either an insertion or a deletion at a given time (in ms, from the start of the translation session) and a cursor position in the text where the action takes place. In the sequence of keystrokes in Figure 5, first the word “`bør `” was written, then it was deleted and replaced by

```
<K time="305229" type="ins" cur="568" str="b"/>
<K time="306187" type="ins" cur="569" str="ø"/>
<K time="306205" type="ins" cur="570" str="r"/>
<K time="306498" type="ins" cur="571" str=" "/>
<K time="308817" type="del" cur="568" str="bør "/>
<K time="308818" type="ins" cur="568" str="m"/>
<K time="308978" type="ins" cur="569" str="å"/>
```

Figure 5. The keyboard pattern represents a replacement of "`bør `" by "`må `". We represent the writing progression in S-notation: "`[bør_` $|_1$`]`^{1}{`må`}" where optionally indexed square brackets represent deletions, curly brackets insertions and the underscore a blank space. The break symbol "$|_1$" indicates an interruption the writing progression at time 1. Note that "`bør_`" was marked and deleted as one block at time `308817`.

```
<K time="307622" type="ins" cur="692" str=" "/>
<K time="307812" type="ins" cur="693" str="m"/>
<K time="308180" type="ins" cur="694" str=" "/>
<K time="308613" type="ins" cur="695" str="d"/>
<K time="309209" type="ins" cur="696" str="e"/>
<K time="309210" type="ins" cur="697" str="r"/>
<K time="309211" type="del" cur="697" str="r"/>
<K time="309343" type="del" cur="696" str="e"/>
<K time="309915" type="del" cur="695" str="d"/>
<K time="310861" type="del" cur="694" str=" "/>
<K time="311147" type="ins" cur="694" str="å"/>
<K time="311299" type="ins" cur="695" str=" "/>
<K time="311481" type="ins" cur="696" str="d"/>
<K time="309209" type="ins" cur="697" str="e"/>
<K time="309210" type="ins" cur="698" str="r"/>
...
```

Figure 6. The example shows a correction pattern "`_m[_der]å_der`" to insert an omitted "`å`". Single keystrokes are used rather than an entire block as in Figure 5. Curly brackets around "{`å_der`}" as well as the interruption symbol in an immediate correction may be left out if they do not increase readability.

```
<K time="52551" type="del" cur="6"  str="nødt "/>
   ...
<K time="55379" type="ins" cur="11" str="nødt "
                refT="52551" refS="6" refE="11"/>
```

Figure 7. This is a delete & paste operation in which the word "`nødt_`" is moved from cursor position 6 to cursor position 11.

"`må `". The caption of the figures also show the writing progression in the S-notation as introduced by Kollberg (1998) and Perrin (2003). Notice that the cursor positions for "`b`" and "`m`" are identical and the deletion operation returns the cursor to the position where it was previously. Figure 6 shows an example where an omitted "`å`" is inserted after "`m`" to produce "`må`" by deleting the first four letters (including a blank space) of the following word "`derfor`" that had already been typed.

Translog also allows the translator to use the clipboard, i.e. to copy a text fragment into a buffer, and to paste it somewhere else in the text later. If the copy operation is linked to a deletion of the fragment from the text (i.e. using ctrl X), the action is represented as a deletion operation, while the pasted fragment is represented as an insertion operation, as shown in Figure 7. In order to keep track of the text's origin, additional information is required with the pasted segment. The time stamp of the copy operation (`time=52551`) serves as a reference together with its starting and end cursor positions. This allows us to trace back keystrokes of the moved segments, as outlined in Section 6.

	A2	K	M	S
insertions	1219	1361	1316	1579
deletions	118	112	261	204

Table 2: Summary of properties for keystroke data, number of insertions and deletions

Table 2 summarises the keyboard activities of the four participants. Fewest keystrokes are counted for subject A2 and most for S. This is not consistent with the length of the produced text, as in Table 1, but correlates with the production time, see Figures 8 and 9.

Before looking into how keystrokes are triangulated with the product data, we shall in the next section present a UAD query language which enables us to retrieve AUs from the product data and to assign it to the relevant process data.

4 A query language

The previous sections show that UAD is represented as a 4-tuple `{S,T,A,K}` for Source and Target texts, Alignment, and Keyboard data respectively. We have implemented a query language to address nodes in the UAD with sets of `attribute.operator.value`. For instance, the expression `[str.=.Although]` matches all nodes in the UAD which contain `str="Although"`. The query language has a number of operators. It allows us to assign variables with the special operator `V`. Sequences of nodes are described by successive square brackets, separated by a comma. The pattern below thus matches two successive nodes and assigns the values of the `time` attribute to the two variables `$T1` and `$T2`.

```
[time.V.$T1],[time.V.$T2]
```

The query language addresses a dimension of the UAD via the elements in `{S,T,A,K}`. Two symbols '`>`' and '`<`' indicate whether a pattern matches forward (from low to high) or backwards. Thus, the pattern in the first line of rule in Table 3 matches every two successive nodes in the keyboard data (`K`), starting from lower to higher time values and assigns the time stamps of the two nodes to the two variables `$T1` and `$T2`. The marker `key` memorises the first node. The construction '`!:`' in the second line interprets and executes the line as a Perl code. In this case it prints out the content of marked node `key` if the time between the two successive keystrokes represented by `$T1` and `$T2` is less than 500ms.

```
1  K>key[time.V.$T1],[time.V.$T2]
2  !:if($T2-$T1<500) {PrintPattern('key');}
```

Table 3. The rule prints out keystroke patterns with less than 500ms between successive keystrokes.

A rule may consists of several successive patterns. In this way several dimensions of the UAD can be correlated and triangulated. Basically, each pattern matches successively every node in the data dimension and instantiates the variables with the retrieved values. If a pattern successfully matches a sequence of nodes, the rule switches to the next line until either a pattern fails or the end of the data was reached. The rule then backtracks and searches for the next pattern in the previous line.

Thus, the rule in Table 4 retrieves all keystrokes that are related to the production of the translation for "developing". The pattern in line `1` iteratively matches all instances of "developing" in the ST and instantiates the variable `$A1` with the cursor positions. The pattern in line `2` retrieves the cursor position of the translation via the alignment data (`A`) and the pattern in line `3` retrieves the word form (e.g. "udviklingslandene") of the translation from the TT (`T`) and stores it in the variable `$V1`. The variables `$A1` and `$A2`

```
1  S>[str.eq.developing,cur.V.$A1]
2  A>[src.eq.$A1,tgt.V.$A2]
3  T>[cur.eq.$A2,str.V.$V1]
4  !:InitKeyRange('Key', $A2-1, $A2+length($V1));
5  K<kp[cur.val.Key]
6  !:PrintPattern('kp:time,type,cur,str');
```

Table 4: The rule prints all keyboard activities related to the translation of “developing”.

thus contain the ST and TT cursor positions of an AU, while the variable `$V1` contains the TT part of the AU.[5]

The keystroke pattern (`K`) in line `5` looks backwards into the keyboard data and marks all keystroke nodes with the marker `kp` which are part of the AU sought. A special operator `val` is used to keep track of the beginning and end cursor positions in the TT segment. The beginning and end cursor positions of the TT segment are initialised with a function:

```
InitKeyRange('Key',$A2-1, $A2+length($V1))
```

`InitKeyRange` initialises the values c_1 and c_2, as described in Section 5, with the beginning position `$A2-1` and the end position `$A2+length($V1)` of the translation for “developing”. The beginning position is set to `$A2-1` to include also the preceding blank space. When going backwards in time through the keyboard data, for each keystroke that modifies the length or position of the translation traced, the values $c_{1,2}$ have to be adjusted. The operator `val` performs this operation as discussed in Section 5.

Finally, the function:

```
PrintPattern('kp:time,type,cur,str');
```

in line `6` prints out the marked keyboard nodes. In this way, the rule retrieves all keyboard activities which belong to the production of the translation for “developing”.

5 Tracing text production

When tracing the production of a TT segment C_i in the process data, we want to retrieve all keystrokes which contribute to the creation of C_i. With

[5]An AU may consist of several tuples `$A1/$A2`, since it may contain more than one word-to-word translation.

an exhaustive fragmentation of the TT into n non-overlapping text segments $C_{1\ldots n}$ every keystroke should be part of at least one C_i.

Assuming we know that text segment C[6] occurs between cursor positions c_1 and c_2 in the final translation, then the length of C equals $c_2 - c_1$. To know what keystrokes contributed to C, we have to look backwards into the logging data and retrieve the appropriate data entries which indicate keyboard activities at position c_i between c_1 and c_2.

Each time an insertion at position c_i is observed, with $c_1 \leq c_i < c_2$, the length of the remaining part of C, which is to be traced, decreases. We thus need to subtract the length of the inserted string at c_i from c_2, so that if c_1 equals c_2 we have found all keystrokes for C and we can stop further search in the keyboard data. In order for this to work, a number of conditions must be met:

- An insertion of length l before cursor position c_1 moves C to the right so that its position was $c_1 := c_1 - l$ and $c_2 := c_2 - l$
- A deletion of length l before cursor position c_1 moves C to the left so that its position was $c_1 := c_1 + l$ and $c_2 := c_2 + l$
- An insertion of l characters after cursor position c_1 and before cursor position c_2 (i.e. a part of C was produced) shortens C so that $c_2 := c_2 - l$
- A deletion of l characters after cursor position c_1 and before cursor position c_2 (i.e. a part of C was deleted) lengthens C so that $c_2 := c_2 + l$

Note that keyboard activities to the right of C, i.e. $c_i > c_2$ can be disregarded.

More case distinctions are necessary if we consider that the length l of the inserted/deleted sequence is longer than 1 and overlaps with the beginning or the end of C, so that $c_i < c_1$ and $c_i + l > c_1$ or $c_i < c_2$ and $c_i + l > c_2$. Updating values for c_1 and c_2 should then only take into account the overlapping inserted/deleted piece, and not the entire length l.

As previously explained in Figure 7, Translog also allows usage of the clipboard to copy & paste sequences of texts. This makes the history of text production arbitrarily complex, since the pasted sequence can, in principle, be recursively composed by pasting together smaller pieces which can be taken from anywhere in the already produced text, or even from outside Translog.

[6]We omit from now on the index $_i$.

For instance, during translation revision, parts of a sentence may be moved and recomposed, word order reversed, and pieces of the text may be partially rewritten etc.

As of now, we have only implemented a tracing operation to follow up text production of the paste operation if the entire sequence C is contained in the pasted segment. That is, for a pasted segment P we assume that $c_1 \geq p_1$ and $c_2 \leq p_2$, where p_1 and p_2 are the start and end cursor positions of P.

Shorter segments C are unlikely to be produced by compositionally pasting their components: one would not usually copy the stem of a word from one part of the text and its inflection ending from another part. However, if C becomes longer, it is certainly possible that C is composed of several pasted sequences. Thus, parts of sentences could be moved or clauses could be restructured by re-using already written material. However, we have not observed such operations in our material so far.[7]

6 Triangulating AUs with keystroke data

AUs are the smallest units which connect the ST and the TT, and keyboard logging data reveals how AUs emerge in time. In Section 5 we have discussed how process data can be assigned to TT segments C. In this section we look at triangulation of AUs with keyboard data. Table 1 in Section 2 shows that almost the entire ST and TT are fragmented into AUs for the four translations, except for a small number of words which remain unaligned. Since we assume that every keyboard activity is driven by the goal to produce a translation of the ST, we can now break the set of keystrokes down into smaller parts and investigate the new properties of AUs independently.

Figures 8 and 9 show how AUs emerge in time. The graphs reveal where there are pauses and deletions, and how keystrokes and gaze activities are distributed over time. It gives a general picture of how the translation develops by relating each activity to the ST unit which it translates. We consider the following keystrokes to be part of an AU:

1. all insertion keystrokes that occur between c_1 and c_2
2. all deletion keystrokes that occur between c_1 and c_2
3. the keystroke immediately preceding c_1

[7]One is more likely to find such patterns in a text production task, as, for instance, writing this text, than in the kind of translation task we are investigating here.

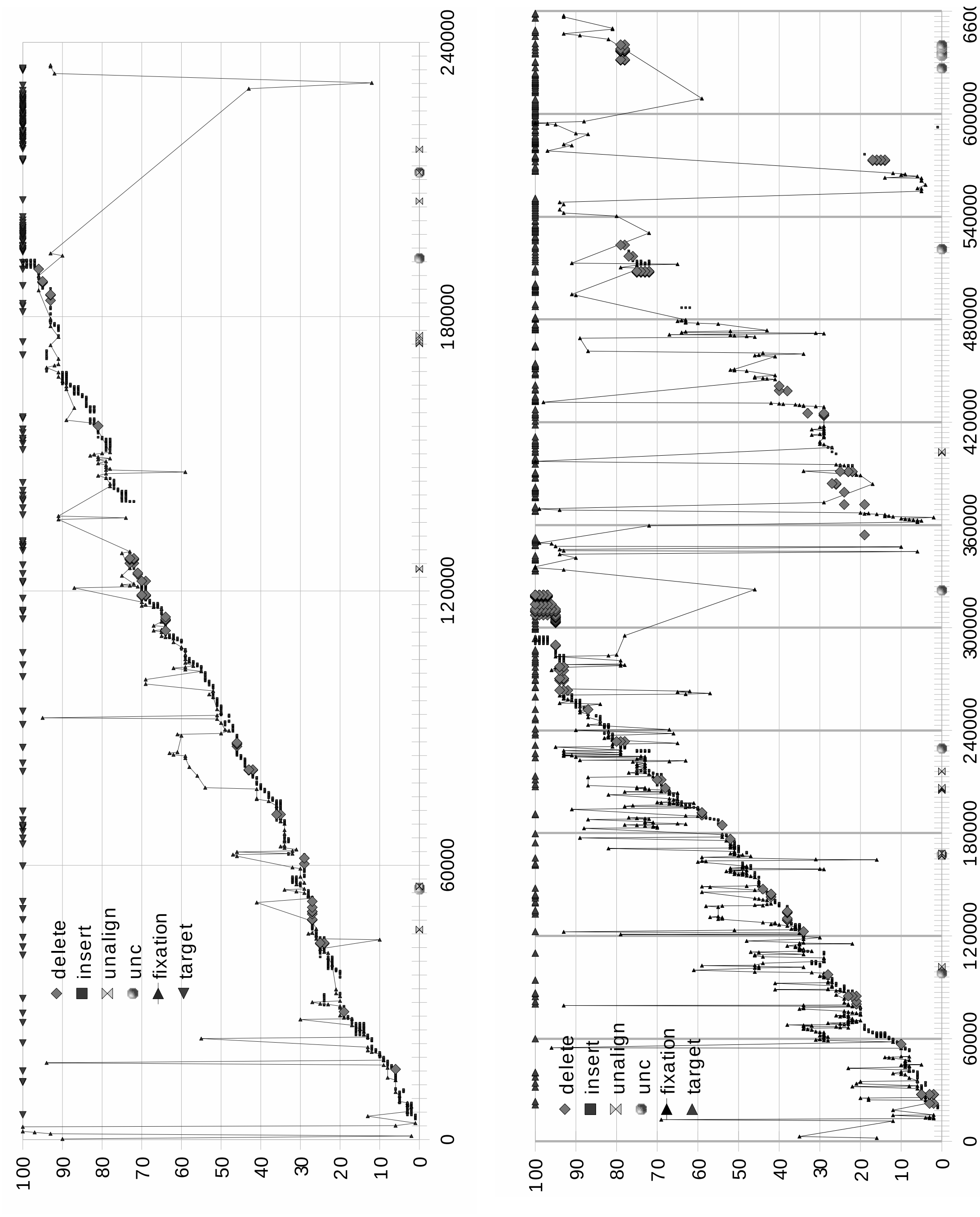
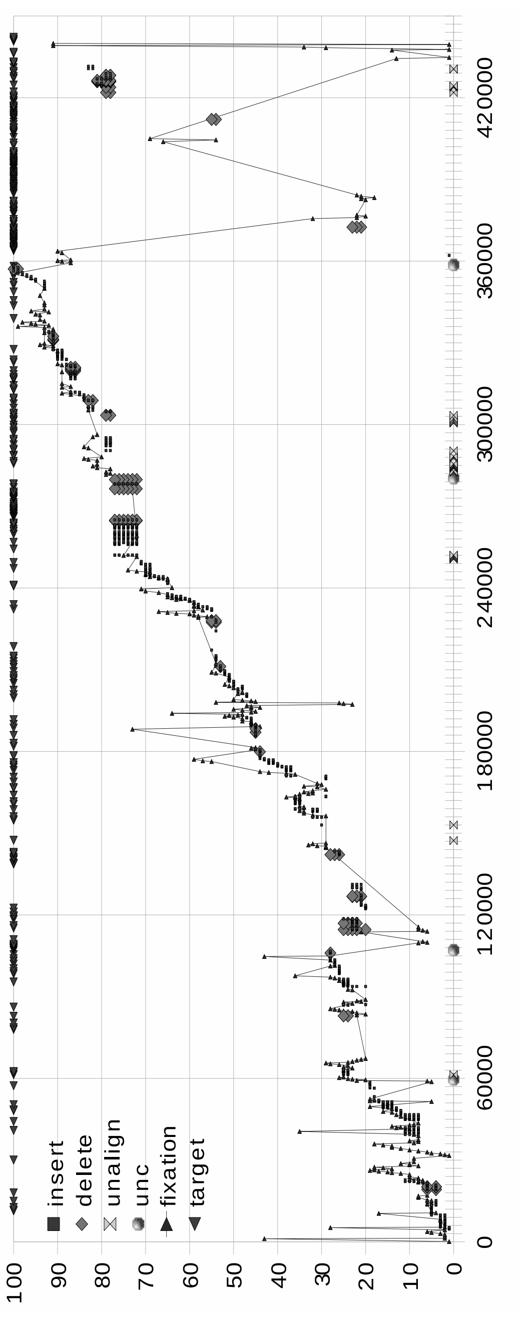


Figure 8. The graphs represent translation progression patterns of 2 subjects, A2 (left) and M (right). On the Y-axis the words of the ST are enumerated (1 to 100). The Y-axis gives production time of the translation in ms. The dots indicate different kinds of keyboard activities (insertions and deletions). The line represents gaze behaviour on the ST words. Fixations on the TT window are indicated with a triangle symbol on line 100. Notice that M needs more than twice as much time as A2, half of it due to revision. The dots on the 0-line indicate keyboard actions which could not be attributed to any AU.



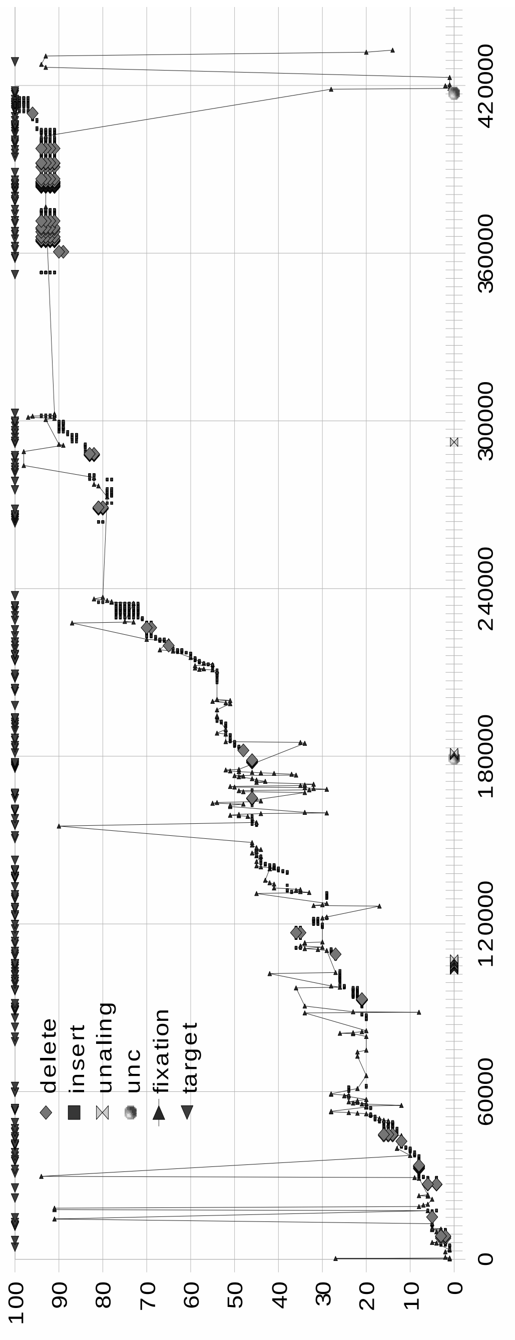


Figure 9. The graphs represent translation progression for K (left) and S (right). Both needed approximately the same time to translate the text. There are long stretches of time in S's translation progression graph where no keyboard and gaze activities are recorded. It is unclear what happened during this period of time.

A) `[udvi]_ud[ivk]vi[lk][l]klingslandene`

	9	`_udviklingslandene`	6	`_udvi`	3	`_udivk`
B)	8	`_udvi`	5	`_udvilk`	2	
	7	`_udvil`	4	`_ud`	1	`udvi`

Table 5. The example plots correction patterns and writing progression of AU$_2$. A) shows the S-notation and B) illustrates 9 correction steps in the development of "`udviklingslandene`".

This definition does not distinguish between types of keyboard actions such as correcting typos, changing inflection or substituting lexemes or even changes as a consequence of reconsidering the syntactic structure of the translation. While such a classification is beyond the scope of this paper, this definition has the advantage that (almost) all keystrokes can be associated with particular AUs.

The example in Table 5 shows the keystrokes involved in the production of the AU$_2$: "`developing countries` $\leftrightarrow$ `udviklingslandene`". The example represents words 2 and 3 of the ST (see Appendix) and the product data is represented in Figures 2, 3 and 4. Table 5 shows a number of (immediate) corrections and their representation in S-notation as outlined in Figure 5: characters in square brackets represent deletions and the underscore "`_`" represents a blank space. Indexes represent temporal order of corrections and identical indexes indicate closeness in time.

Table 5 shows the development of "`udviklingslandene`", looking backwards into the keyboard data. It consists of 9 steps, of which the last corresponds to the final form as observed in the final translation. In steps 5 to 8 the characters "`kl`" are accidentally reversed and corrected. Steps 3 and 4 show a similar procedure with correction of reversed letters "`vi`". At step 2, the entire sequence "`udvi`" is deleted, in order to insert a blank space before the word which had been omitted in step 1.

We incorporate steps 2 and 1 into AU$_2$, since we consider the keystroke immediately preceding the TT sequence to be part of the AU. Otherwise these keystrokes could not be assigned to any AU. For the same reason all the keystrokes in Figure 5 are part of the same AU, and the keystrokes "`_m[_der]å`" in Figure 6 are part of another AU.

A more complex keyboard pattern can be observed during the production of AU "`go` $\leftrightarrow$ `at iværksætte`" for word number 38 of the ST in the Appendix. The example is taken from translator M (Figure 8) where 36

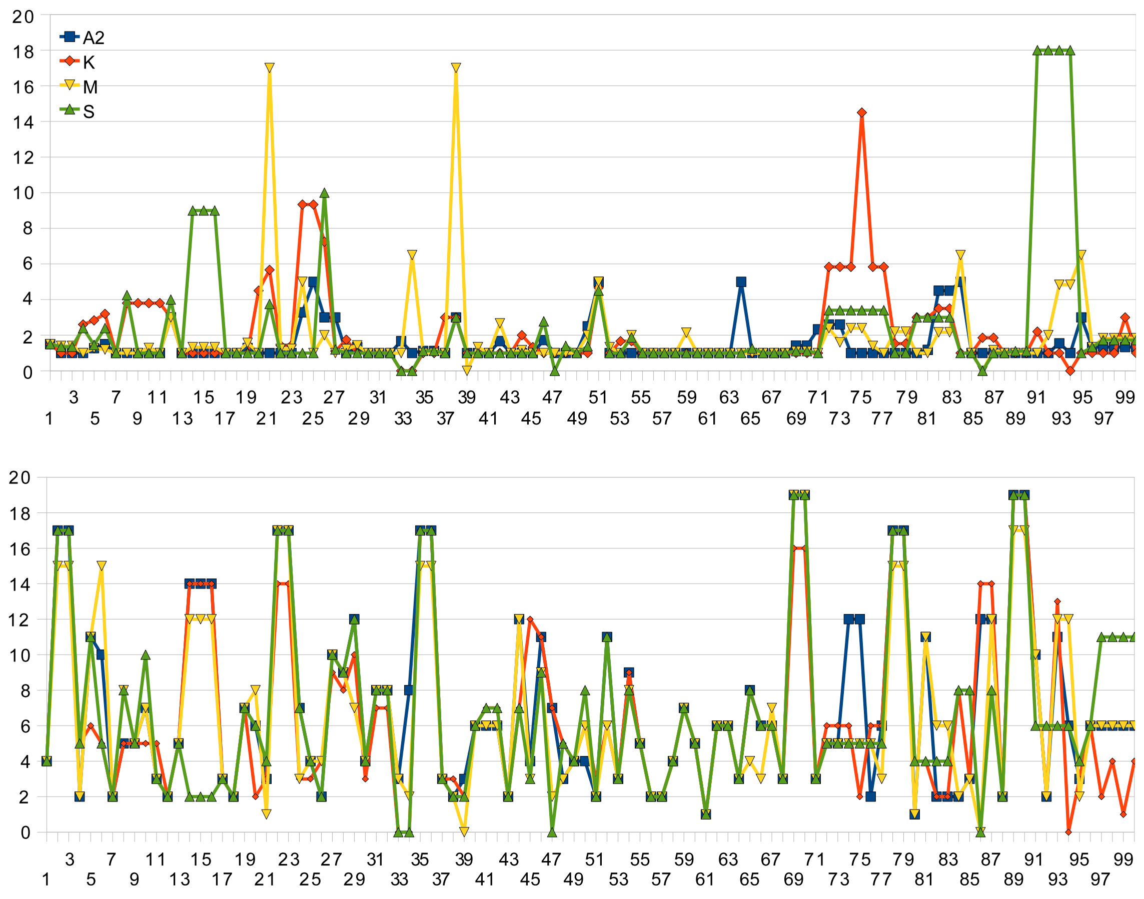


Figure 10. These graphs represent properties of the AUs and their associated keyboard activities for the four subjects $\{A2, K, M, S\}$. The horizontal X-axis lists the words in the ST. The vertical Y-axis in the top graph shows the number of $keystrokes/length$ of the AU translation. A value of 1 means that the length of the AU translation equals the number of keystrokes (i.e. no corrections take place during the production of that AU). A value > 1 indicates deletions. For instance, at word number 21 (translation of "on"), translator M needs 17 times more keystrokes than the length of the translation.

The bottom figure plots the length of the translation for each ST word. It shows, for instance, that the translation of word 21 (in the M data) is one character long. It also shows that translations of most words have similar length, across the four translations. Note that some of the divergences might be due to inconsistencies in alignment strategies, e.g. words 7 to 11 and 95 to 99.

AUs can be recognised through the same length and/or number of keystrokes. For instance, ST words 14 to 16 are part of the same AU. In each translation they have the same length (bottom) and the same number of keystrokes (top), which indicates that these words are clustered into one AU.

keystrokes are counted (including deletions) to produce the translation of 13 characters:

_[ta_$|_1$]^1 at_[gøre_$|_2$]^2 iværksæ[kk]$^3|_4$ {tt}^{4} e$|_3$

As introduced in Figure 5 Kollberg (1998) uses the symbol "$|_i$" to indicate an interruption at time i. A distinction is made between i) *insertion interruptions*, in which case the interruption symbol is placed immediately to the right of the last insterted character, as in "$|_{1,2,3}$" above and ii) *revision interruptions*, in which case the symbol is immediately to the right of the closing revision bracket, as shown in "$|_4$" in the example above. Each interruption refers to a co-indexed insertion or deletion action.

The pattern starts with the correction of "ta" into "at" followed by the typing and deletion of "gøre", presumably due to a reconsideration of the syntactic structure. Then "iværksække" is produced and finally "kk" is corrected into "tt". These activities take place at different points during the translation: while the first two corrections take place while typing the words, the substitution "[kk]$^3|_4${tt}^{4}" is represented in two successive actions, a deletion of "kk" at revision time $|_3$ which is immediadely followed by the insertion of "tt" at time $|_4$.

Figures 8 and 9 better reveal the temporal distribution of keyboard actions. Particularly M (Figure 8) has a very long post-editing phase, which shows that some AUs emerge in different sequences of time.

Some of the keyboard activities cannot be attributed to any AU. There are two reasons for this: i) a word in the TT has not been aligned to any ST word. This is mainly the case for punctuation marks, but also sometimes for determiners or (relative) pronouns like Danish "den", "der" which cannot (or are not) connected to any ST word(s). Subsequently those keyboard activities do not belong to any AU. Another reason is ii) the presence of sequences of insertions and deletions immediately before such unaligned items. Table 6 summarises these keystrokes for the four participants.

	A2	K	M	S
unclassified	32	57	24	33
unaligned	3	4	21	2

Table 6: The number of unaligned and unclassified keystrokes in the UAD.

For instance, the following keystrokes are part of such unclassified activities. The segment was first inserted at the end of the text, then modified several times (as can be seen through the square brackets) and finally replaced by “`følges ad`”. It is the last AU from the data of M in Figure 8:

```
[bliver_derfor_nødt_til[at_]_a_gå_hånd_i_hån]
```

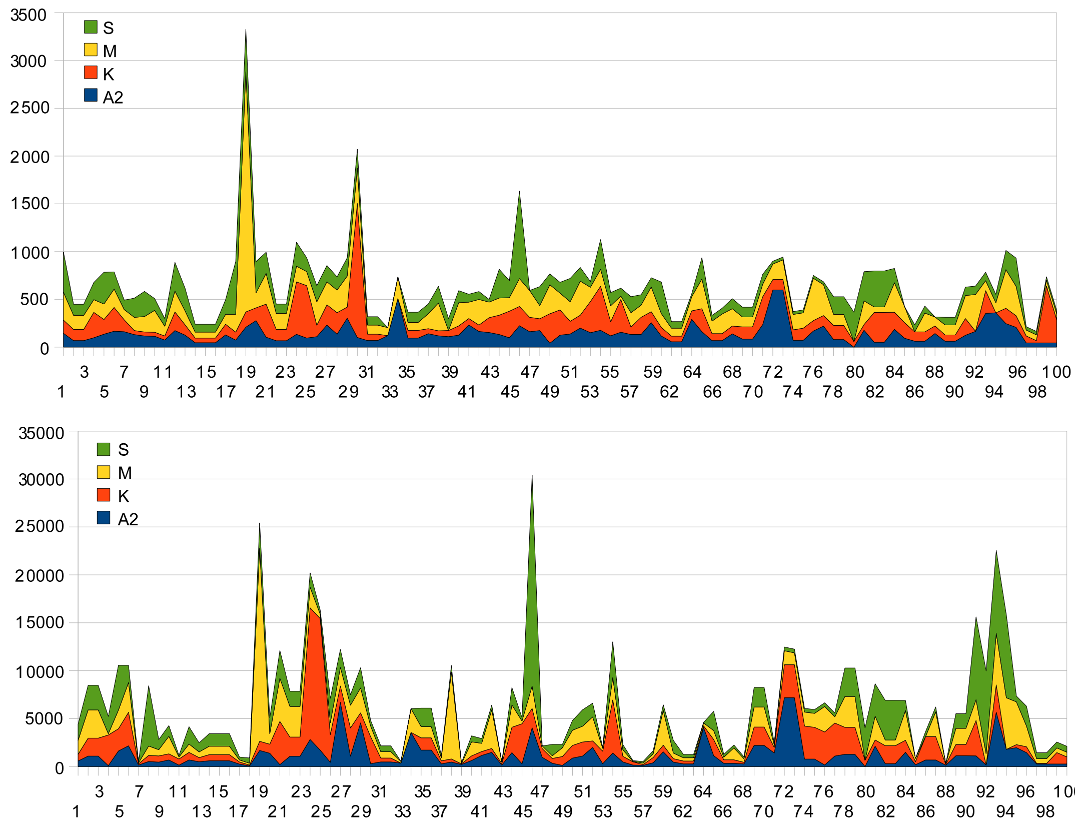


Figure 11. These graphs show the time delay between successive keyboard activities of the four subjects. The X-axis plots the words in the ST. The Y-axis in the top Figure shows average intra-word keystroke time span between two successive keystrokes within the production of one AU. The bottom Figure shows inter-word keystroke time delay between the end of an AU and the start of typing of the next AUs. Notice that the time scale is one magnitude higher for inter-AU pauses than for intra-AU pauses.

The graphs in Figures 10 and 11 resume the quantification of AUs. Figure 10 puts into relation the length of the AU and the number of keystrokes, while Figure 11 puts into relation the time between successive keystrokes between and within words.

It is interesting to notice that there seems to be a difficult part in the translation of words 19 to 30, words 71 to 85 and 90 to 99. This is particularly visible in the $keystoke/length$ ratio in the top graph of Figure 10, where many AUs are produced by all translators using more than 5 times the keystrokes of the length of the final translation. It is obvious that there are parts which are easy for all translators, i.e. where the $keystoke/length$ ratio is 1 for everyone, and that there are parts which are 'difficult' for everyone. A similar, although not so clear picture emerges in the inter- and intra-word keystroke time span in Figure 11. Also here, much time is spent during the translation of words 19 to 30, 71 to 85 and 91 to 99, particularly on inter-word time spans as shown in the lower graph, but other factors seem to interfere. For instance, much time is also spent around the translations of word 45, for both inter- and intra-word keystrokes, but this obviously did not have a high impact on correction patterns (Figure 11). We hope to investigate this further and to relate these quantitative findings to the linguistic properties of the AU.

7 Discussion

Krings (2005) suggests an "ascending abstraction" of categories in which translation process research should be carried out. The basis of research is an analysis of the phenomena which is "so to speak the degree zero of data analysis"(my transl.). This basic investigation should be followed by a classification, quantification and correlation of the phenomena, and subsequently by the detection of causal relations, which finally leads to the elaboration of a theory. According to him (as of 2005) most translation process research either gets stuck in listing phenomena or in classifying them, and sometimes reaches the level of quantification. Given the novelty of the area of research, this is, according to Krings, not surprising.

With this work, we seek to push research further into the investigation of correlations, the systematical investigation into the statistical dependence between two (or more) variables. To do so, we need a consistent data structure in which the data is represented. We also need a reasonable amount of data on which statistics can be based. Further, we need a sufficiently expressive formalism with which we can formulate the variables that we are interested in and an automated method to query the data so as to retrieve all instances of patterns which satisfy the description of these variables. Only then can we quantify dependencies in the data and eventually detect degrees of causal relations.

This paper has outlined a strategy and a set of tools ready to use for cross-validation and triangulation of Translog product and process data. Besides the key-logging data, Translog also provides two texts, the ST and the

TT. The notion of Alignment Unit (AU) allows us to fragment the product data into smaller segments which can be quantified and triangulated with process data. While a general theory of human translation founded on translation process data may perhaps not be reached in the near future, we show a way of correlating and modelling the data in a quantitative manner. At this point we are still far from being able to formulate conditional probabilities over Process Data (PD) and AUs (and maybe also Translation Units) which would answer questions such as: given a history of PD what is the probability of the next AU$_i$ to be deleted, inserted or modified? The extent to which we can answer this and related questions will ultimately determine the success of integrating advanced translation aids with human translation activities.

Flick (2008, 20) points out that triangulation is to be understood as a “strategy on the way to a deeper understanding of the investigated object and thus a step towards more knowledge and to a lesser extent to validity and objectivity of the interpretation” (my trans.). And so we hope that triangulating product and process data may contribute to a deeper understanding of human translation processes.

References

Alves, F. and Vale D.C. 2009. Probing the Unit of Translationin time: aspects of the design and development of a web application for storing, annotating and querying translation process data. *Across Language and Cultures*, forthcoming.

Carl, M. and Jakobsen A.L. 2009. Towards statistical modelling of translator's activity data. *International Journal of Speech Technology*, forthcoming.

Carl, M. Jakobsen, A.L. and Jensen, K.T.H. . 2008. Studying human translator behavior with user-activity data. In *Natural Language Processing and Cognitive Science (NLPCS 2008)*, Barcelona, Spain.

Dagan, I., Church, K.W. and Gale, W.A. 1993. Robust bilingual word alignment for machine aided translation. In *Workshop On Very Large Corpora: Academic And Industrial Perspectives*.

Dragsted, B. 2004. Segmentation in translation and translation memory systems. Dissertation, Copenhagen Business School.

Flick, U.. 2008. *Triangulation Eine Einführung*, volume 12 of *Qualitative Sozialforschung*. VS Verlag für Sozialwissenschaften, Wiesbaden.

Hansen, G. editor. 1999. *Probing the Process in Translation: Methods and Results*, volume 24. Copenhagen: Samfundslitteratur.

Jakobsen, A.L. 1999. Logging target text production with Translog. In *(Hansen 1999)*, pages 9–20.

Jakobsen, A.L. 2006. Research methods in translation - Translog. In *(Sullivan and Lindgren 2006)*, pages 95–105.

Kollberg, P. 1998. S-notation — A Computer-based Method for Studying and Representing Text Composition. Licenciate thesis, Royal Institute of Technology, Stockholm.

Krings, H.P. 2005. Wege ins Labyrinth - Fragestellungen und Methoden der Übersetzungsforschung im Überblick. *Meta*, 50(2):342–358.

Kromann, M.T. 2003. The Danish dependency treebank and the DTAG treebank tool. In *2nd Workshop on Treebanks and Linguistic Theories*, Vaxjo, Sweden.

Och, F.J. and Ney, H. 2000. Improved Statistical Alignment Models. In *ACL*, pages 440–447, Hongkong, China.

Perrin, D. 2003. Progression analysis (PA): investigating writing strategies at the workplace. *Pragmatics*, 35:907–921.

Sullivan, K.H.P. and Lindgren, E. editors. 2006. *Computer Keystroke Logging and Writing: Methods and Applications*, volume 18. Elsevier, Oxford.

Appendix

The ST consists of the following 100 words. For technical reasons punctuation marks were deleted from the text but are shown here in brackets (·) to facilitate reading:

1	Although	26	threaten	51	minimise	76	of
2	developing	27	economic	52	emissions	77	it (.)
3	countries	28	development (.)	53	from	78	Developing
4	are	29	Incentives	54	deforestation (.)	79	countries (,)
5	understandably	30	must	55	Some	80	in
6	reluctant	31	be	56	of	81	particular (,)
7	to	32	offered	57	the	82	need
8	compromise	33	to	58	most	83	to
9	their	34	encourage	59	vulnerable	84	adapt
10	chances	35	developing	60	countries	85	to
11	of	36	countries	61	of	86	the
12	achieving	37	to	62	the	87	effects
13	better	38	go	63	world	88	of
14	standards	39	the	64	have	89	climate
15	of	40	extra	65	contributed	90	change (.)
16	living	41	green	66	the	91	Adaptation
17	for	42	mile	67	least	92	and
18	the	43	and	68	to	93	mitigation
19	poor (,)	44	implement	69	climate	94	efforts
20	action	45	clean	70	change (,)	95	must
21	on	46	technologies (,)	71	but	96	therefore
22	climate	47	and	72	are	97	go
23	change	48	could	73	bearing	98	hand
24	need	49	also	74	the	99	in
25	not	50	help	75	brunt	100	hand (.)

This is an empty page (however, for technical reasons it cannot be entirely empty)