Automatic recognition of linguistic replacements in text series generated from keystroke logs

Daniel Couto-Vale, Stella Neumann, Paula Niemietz
RWTH Aachen University, Germany
Kármánstr. 17-19, D-52062 Aachen
danielvale@icloud.com, {firstname.familyname}@ifaar.rwth-aachen.de

Abstract

This paper introduces a toolkit used for the purpose of detecting replacements of different grammatical and semantic structures in ongoing text production logged as a chronological series of computer interaction events (so-called keystroke logs). The specific case we use involves human translations where replacements can be indicative of translator behaviour that leads to specific features of translations that distinguish them from non-translated texts. The toolkit uses a novel CCG chart parser customised so as to recognise grammatical words independently of space and punctuation boundaries. On the basis of the linguistic analysis, structures in different versions of the target text are compared and classified as potential equivalents of the same source text segment by 'equivalence judges'. In that way, replacements of grammatical and semantic structures can be detected. Beyond the specific task at hand the approach will also be useful for the analysis of other types of spaceless text such as Twitter hashtags and texts in agglutinative or spaceless languages like Finnish or Chinese.

Keywords: translation process, keystroke logging, character chart parser

1. Introduction

In a naïve approach, it would be assumed that the task of translating consists in giving literal renditions of the words of the source language text in the target language, i.e. in finding equivalent words for each word in the source text and simply replacing the source text words by these target language words in the same order. However, various factors such as the incommensurability of the languages involved, differences in the situational features that determine a register, specific goals and guidelines of the translation brief among others will result in non-literal translations.

Moreover, the task of mediating between two potentially different lingua-cultures also leads to translations being different in terms of the distribution of linguistic features from original text production in the target language in several ways (Hansen-Schirra et al., 2012). Machine learning approaches to translation as a sub-language ('translationese') report high accuracies in classifying texts as translations and non-translated texts in a mixed set based on properties such as mean 1-gram length, frequencies of spellings of derivational and inflectional prefixes and suffixes, ratio between function words and content words (Volansky et al., 2015). While such studies provide evidence for the claim that translations are indeed different from non-translated texts, they do not easily support explanations for the observed differences. It is also assumed that some of the statistical patterns are caused not by linguistic and cultural differences, but by cognitive factors such as (lack of) understanding of the source text and fatigue by the translator.

Process-oriented translation research aims at characterising this cognitive activity in terms of translators' behaviour during the translation task. For this purpose, keystroke loggers such as Translog (Jakobsen and Schou, 1999; Carl, 2012) are used to record and replay the text-producing actions of different translators in the task of translating the same source text from language A to language B. Instead of analysing corpora of the products of the translation process in the carefully composed final order in which they are deemed to best achieve their goals, log files represent the text production process as a chronological series of computer interaction events (log events such as character key strokes and hot key strokes, mouse clicks etc.). A text production process will always include sequences of uninterrupted text production, but it may also include (i) potentially discontinuous modifications (insertions, deletions) of text produced so far when an existing part of the text is modified at a later stage and (ii) actions of changing cursor positions, in which case insertions in the middle of the produced text move the tail of the text right and down to other screen coordinates. Especially in the revision phase towards the end of the text production process, modifications consisting of individual log events occur at various places of the text produced so far.

Processual data enables researchers not only to align source text segments with segments of the final target text (**?**, 'equivalence', cf)]Catford:1965vc, but also to track the series of possible alternate equivalents produced by one and the same translator for the same original segment throughout time. Intermediate (non-final) target texts offer a unique opportunity for researchers to observe intermediary stages of a chained transfer procedure and also to infer personal and collective preferences between divergent equivalents when translators consider two or more alternates.

Keyloggers are useful for collection of such data since they store timestamped text-producing actions by the translator. Figure 1 illustrates an action of replacing the underlined part of the nominal group *eines solchen Balls Papier* ('of such a ball of paper') by the more frequent wording *eines solchen Papierballs* ('of such a paper ball'). This replacement actually consists of backwards deleting 12 individual characters from lines 1622 to 1633 and inserting 11 characters from lines 1634 to 1644 in the XML representation of the log events.

The human analyst interested in understanding which textual units attract particular attention (and which ones do

Figure 1: Log events in Translog XML (lines 1620-1650 out of 4072).

not) is faced with the task of reassembling such individual log events (see Figure 1) into strings of letters of unfolding words for further analysis of, say, which linguistic elements are replaced and what they are replaced by. Though log files are comprehensive and very useful, manually analysing such data is at times cumbersome and at times altogether impossible for the human analyst. Among the reasons why analysis is so difficult is the fact that translators' actions during text production affect an intermediary text produced until then that the analyst does not have access to visually and must reconstruct mentally from the beginning of the file.

In this paper we present a method for creating a human-inspectable visualisation of keystroke logs in a corpus of keystroke logging files and of automatically analysing them linguistically, which allows us to detect replacements of grammatical and semantic structures between different text versions automatically. The remainder of the paper is organised as follows. Section 2 will elaborate on the role that replacements play in translation and why a computational tool is useful for analysing them. The toolkit developed for the analysis of replacements is introduced in two steps: In Section 3 the approach to parsing log files is explained, while Section 4 discusses the approach to detecting alternate translation attempts in the log files. Section 5 provides details on the evaluation of the toolkit we developed. Section 6 discusses the limitations of our toolkit, and, finally, Section 7 summarises the main findings and provides an outlook on future work.

2. The role of replacements in translation

Experimental studies have shown that translators approach their task in a cyclic fashion, not simply translating one source text segment by a final target text segment, but rather attempting a translation and coming back to it for revision potentially at various times during the translation process (Carl, 2009; Alves and Couto-Vale, 2011). Especially in cases when the translator finds it difficult to map lexical, grammatical or semantic structures between the languages, the translation may be attempted in several ways involving replacements of previously written segments by new ones. For instance, this can involve packing phenomena represented by a clause in an earlier translation attempt into a segment represented by a nominal group, a strategy that will often include nominalisations. When we detect such a replacement, we gather evidence that some translations might be divided into a first transfer procedure between similar verbal representations in each language (clause to clause, nominal group to nominal group) and a second procedure between representations of different ranks in the same language. This means that a rank shift might happen between two target representations and not necessarily between source and target ones for the translation of some segments. It is equally conceivable that the first transfer procedure already involved a rank shift across languages (nominal group to clause) with the second procedure involving another rank shift (clause to nominal group) resulting in a final translation that appears inconspicuous when only comparing the source text with the final translation product.

Steps of translations such as these can be reconstructed from logged text-affecting actions. This type of information may yield insight among others into how specific features of translations come about. Rank shifts and other translational phenomena have been discussed for a long time (Vinay and Darbelnet, 1958). More recently, Cyrus (2006) and Čulo et al. (2008) provided corpus evidence that transfer procedures have a residual effect on translated texts. In addition, a better understanding of such procedures has also been recognised as a bottleneck for machine translation (Helmreich et al., 2004). Therefore, modelling how humans translate may result not only in a theory of translation better capable of describing, explaining, and predicting translational phenomena but also in a better understanding of what can be done to improve machine translation.

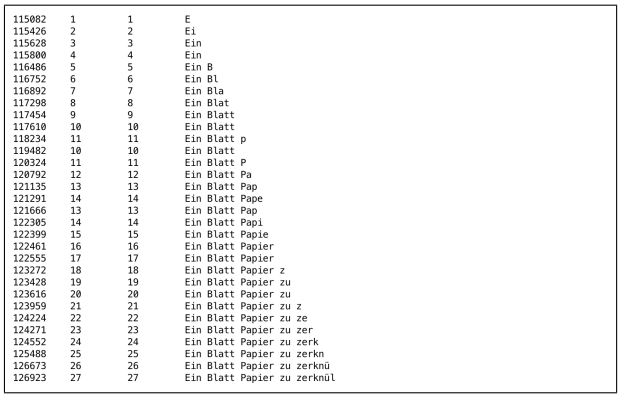

Figure 2: Beginning of a text series file

Keystroke logs contain detailed timestamped descriptions of translators' sequential text-affecting actions such as 'pressing the P key while the shift key is held down and while the cursor is at position 110 in the text' (line 1634 in Figure 1). However, since characters are inserted at the cursor position (not always at the end of the text written so far) and since inserted characters may replace selected text segments, analysing keystroke logs linguistically through inspection is difficult even for the most elementary tasks such as that of recognising inserted words. For the harder tasks of recognising insertions and replacements of grammatical and semantic structures, which are where new insights are to be achieved, it becomes altogether impossible.

A computational tool should help detect and visualise keystrokes on related text-affecting actions to support further analysis. We demonstrate such a tool based on analysing a keystroke logged corpus collected by Serbina et al. (2015). The corpus contains the logs generated with Translog v. 3.0.1 based on 16 translations of a popular-scientific text about the physical properties of the action of crumpling paper.[1]

The computational task involves, among other things, identifying meaningful grammatical units from a not necessarily meaningful stream of log events that may include keystrokes in an inconsistent order from the linguistic point of view as well as identifying versions of the ongoing text production that are actually modifications of the same element (e.g. the replacement of *Balls Papier* by *Papierballs* in Figure 1) and not chance occurrences of similar lexico-grammatical items that are to be expected to occur multiple times in a text on a given topic. Our solution to this task involves drawing on grammatical analysis already in the initial task of identifying meaningful units in the intermediary versions of the target text.

Drawing on a computational solution to the problem laid out here also has an additional advantage. Since the units that the parser recognises (see Section 3) are not necessarily unambiguous, the parser will produce several alternative analyses (a standard approach to handling ambiguous items in language data). This is seen as a major advantage for the purposes of the present analysis, because the ambiguity of the text production data is intended to be kept in the linguistic analysis. In many cases, it will be impossible to determine which of the possible readings of an incomplete textual unit was intended by the author. Drawing on an approach to a similar problem in learner corpus research (Lüdeling, 2008), the various readings are captured as target hypotheses, i.e. the analyst's different interpretations of the target construction the learner/writer attempted to use, rather than deliberately narrowing down the possible readings to the one the analyst happens to prefer.

3. How to parse texts

For recognising replacements, we first developed a software[2] for creating a **text series file** for each log file. Each line of these files contains four tab separated values: a timestamp of a text-producing action and the resulting dot position, mark position,[3] and text (see Figure 2).

Because these files are too large for any thorough manual annotation, we used a parser for annotating all text versions. Since we were interested in replacements such as *Ball Papier* ('ball of paper' in German) by *Papierball* ('paper ball'), we could not rely on gram-based chart parsers. For that reason, we customised the OpenCCG chart parser (Steedman and Baldridge, 2011) so that it is capable of recognising grammatical words independent of space and punctuation boundaries.[4] In particular, this parser is capable of recognising *Papier* and *ball* as two different words even when they are written together as in *Papierball*.

Figure 3 illustrates an automatically recognised replacement of a 2-gram by a 1-gram. The first segment highlighted in pink is the replaced 2-gram and the second one highlighted in purple is the substitute 1-gram.

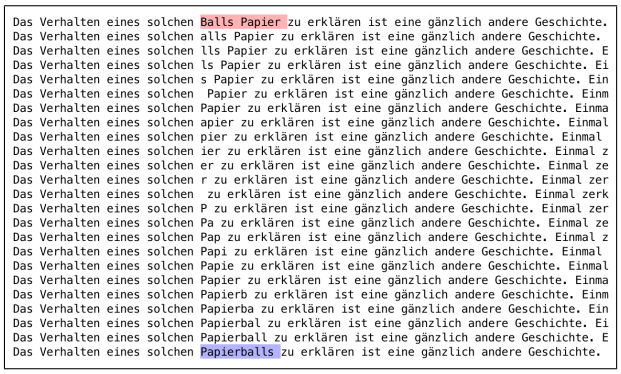

Figure 3: Replacement of *Balls Papier* by *Papierballs* automatically recognised and highlighted in our application

Our new method of parsing does not treat all space-separated letter sequences (1-grams or orthographic words[5]) as word spellings. By separating the notion of 1-gram from that of word spelling, we were able to consider any sequence of morphemes a separate grammatical word according to our needs. As a result, we could treat both the 1-gram *Papierballs* 'paper ball' and the 2-gram *Balls Papier* 'ball of paper' as a sequence of two grammatical words. This way of cutting strings (see Section 4) into word spellings and not grams shall facilitate the next step of judging whether two wordings are alternate equivalents.

eines∗	solchen∗	Balls∗	Papier∗
Modifier	Modifier_2	Head	Modifier
–	–	Thing	Material

Table 1: A 4-gram with 4 grammatical words

eines∗	solchen∗	Papier	balls∗
Modifier	Modifier_2	Modifier	Head
–	–	Material	Thing

Table 2: A 3-gram with 4 grammatical words

In Tables 1 and 2, the symbol ∗ represents a space 'inside' a word spelling, the last character of a word spelling being usually a space. However, since word spellings are not necessarily 1-grams, not all word spellings recognised by our parser end in a space character. Some of them such as *Papier* in the nominal group *eines∗solchen∗Papierballs∗* end with a letter, in this case the letter *r*.

The process for achieving this is the following. First an $n \times n$ matrix is created where n is the number of characters in the text string. Each $\text{cell}_{i,j}$ of the matrix represents

[1] One file had to be excluded from our analysis due to poor translation quality. Another file had to be discarded at a later stage due to incomplete logging of mouse interactions by Translog (see Section 5).

[2] Software available at https://github.com/DanielCoutoVale/TranslogToolbar

[3] Documentation: https://docs.oracle.com/javase/tutorial/uiswing/events/caretlistener.html

[4] Parser available at https://github.com/DanielCoutoVale/openccg

[5] A word is a sequence of n orthographic words.

a substring$_{i,\,j}$ where i is the initial character of the string and j is the last. Because i must be lesser or equal j only half of the matrix can be filled. Matrices of this kind are called 'charts' in parsing. The difference between our chart parser and traditional chart parsers is that we use a 'character chart' whereas other approaches use '1-gram charts'.
The first module that fills the chart is a word spelling recogniser. This module recognises all word spellings that are predicted with an internal vocabulary. In our case, we created the internal vocabulary based on a corpus of the final target texts by translators. Since we have several translations of the same original text, the coverage of the word spelling recogniser is close to 100% for intermediate versions.
From that point on, a combinatory categorial unifier takes words as instances of 'combinatory categories' and unites them according to the combinatory rules that apply to each structure category. Two grammatical structures categorised according to combinatory rules are only to be united if the spelling of the first ends immediately before the spelling of the second and if the combinatory rules allow it. Each composite structure produced by the parser has a spelling resulting from the concatenation of the spellings of its parts. Each composite structure is put into a chart cell that corresponds to its spellings. Figure 4 shows a filled character chart for the 2-gram *ein Papierball* 'a paper ball'. Notice that both the incomplete structure for the substring$_{1\,-\,10}$ and that for the substring $_{5\,-\,15}$ are inserted. The complete nominal group for the substring $_{1\,-\,15}$ is inserted into the chart as expected.

e	i	n	•	P	a	p	i	e	r	b	a	l	l	•
1–1	1–2	1–3	1–4	1–5	1–6	1–7	1–8	1–9	1–10	1–11	1–12	1–13	1–14	1–15
	2–2	2–3	2–4	2–5	2–6	2–7	2–8	2–9	2–10	2–11	2–12	2–13	2–14	2–15
		3–3	3–4	3–5	3–6	3–7	3–8	3–9	3–10	3–11	3–12	3–13	3–14	3–15
			4–4	4–5	4–6	4–7	4–8	4–9	4–10	4–11	4–12	4–13	4–14	4–15
				5–5	5–6	5–7	5–8	5–9	5–10	5–11	5–12	5–13	5–14	5–15
					6–6	6–7	6–8	6–9	6–10	6–11	6–12	6–13	6–14	6–15
						7–7	7–8	7–9	7–10	7–11	7–12	7–13	7–14	7–15
							8–8	8–9	8–10	8–11	8–12	8–13	8–14	8–15
								9–9	9–10	9–11	9–12	9–13	9–14	9–15
									10–10	10–11	10–12	10–13	10–14	10–15
										11–11	11–12	11–13	11–14	11–15
											12–12	12–13	12–14	12–15
												13–13	13–14	13–15
													14–14	14–15
														15–15

Figure 4: A character chart filled with complete and incomplete grammatical structures.

Since our parser fills a character chart by recognising word spellings and uniting structures based on their combinatory categories, it does not depend on a preprocessing with a 1-gram tokeniser nor a part-of-speech tagger (pos-tagger). As a result, both grammatical structures *eines*∗*solchen*∗*Balls*∗*Papier*∗ in Table 1 and *eines*∗*solchen*∗*Papierballs*∗ in Table 2 could be treated by our parser as nominal groups directly composed of four words with no intermediary unit corresponding to the 1-gram *Papierballs*∗. In other words, our parser does not need to take 1-grams as grammatical atoms from a preprocessing phase nor to build up any structure that corresponds to them in order to complete a linguistic analysis. In that sense, our parser does not distinguishes the two structures in Tables 1 and 2 but by the fact that they differ in word order and that the chosen word spellings are also different (*Balls*∗ vs *balls*∗ and *Papier*∗ vs *Papier*).
Once parsing is finished, a semantic structure is created in terms of what nominal groups represent. Tables 3-5 are examples of such semantic structures. The prefix *eum:* stands for *Experiential Upper Model* (Bateman et al., 2010) whereas the prefix *deu:* stands for *Deutsches Unteres Modell* 'German Lower Model'.

Superclass	Subclass
eum:Thing	deu:Ball
eum:Material	deu:Papier

Table 3: *eines solchen Balls Papier* as representation

Superclass	Subclass
eum:Thing	deu:Ball
eum:Material	deu:Papier

Table 4: *eines solchen Papierballs* as representation

Superclass	Subclass
eum:Thing	deu:Ball
eum:Material	deu:Papier

Table 5: *der Papierball* as representation

Superclass	Subclass
eum:Thing	deu:Kugel
eum:Material	deu:Papier

Table 6: *eine Papierkugel* as representation

Contrary to grammatical structures, two representations such as those in Tables 3 and 4 can be identical even if the word sequences are different. Moreover, two representations such as those in Tables 4 and 5 can also be identical despite the fact that the grammatical case and the deictic terms of the nominal groups are different. Furthermore, two representations such as those in Tables 5 and 6 can be alternate equivalents of the same original during a translation even if they differ as far as representation is concerned. All classes of phenomena specified in a general upper model such as things and materials are further specified with language specific classes of representable phenomena (taxa). These general and specific classifications of representable phenomena are what enables a system to detect alternates as we shall demonstrate in the next section.

4. How to detect alternates

We implemented a procedure to 'difference' the parse charts of each pair of consecutive target text versions. It generates a 'chart difference' for each pair in the same way as version control systems do for different file versions, in our case keeping track of insertions and deletions of lexicogrammatical and semantic structures from a chart to the next. By comparing two consecutive charts, we detect which grammatical structures are present in only one of the

two charts, indicating that a structure was added to or removed from the unfolding translation. For example, each time the second text string is one character shorter than the first, the procedure searches for all the structures that exist in the first chart either at the same position as in the second chart or one position before. If the structure is not found there, then it is considered removed.

A second procedure iterates over the chart difference series to find replacements. Whenever it finds a removal of a grammatical structure, it searches for additions of structures in the 50 following differences in a window of 25 characters from the dot position. If it finds an addition of structure, it sends the removed-added structure pair to software modules that we called 'equivalence judges'.

For each segment of text, the parser builds zero or more lexicogrammatical structures, each one associated with the corresponding semantic structures in the form of entities. When a removed grammatical structure is compared with an added structure for equivalence, the 'equivalence judges' are activated. If the structures in the pair are judged by any of the judges to be potential equivalents of the same segment in the source text, they are kept in the list of replacements.

Two equivalence judges have been implemented so far. The first equivalence judge checks whether two references to an entity are references to instances of the same type. For instance, *eines solchen Balls Papier* ('such a ball of paper') and *eines solchen Papierballs* ('such a paper ball') are instances of the same class of things made of the same material. Although lexicogrammatically different, these entities are identical as far as their semantics is concerned. The second equivalence judge checks whether two references to an entity are references to instances of different but interchangeable types in a given situation. For instance, *ein zerknülltes Blatt Papier* ('a crumpled sheet of paper') and *eine Papierkugel* ('a paper ball') are indeed two ways of classifying things, but they happen to be references to the same thing and are in the same textual position. In that sense, the second equivalence judge checks whether two entity classes are interchangeable ways of classifying the same thing in a given situation. This equivalence judge relies on user-specified groups of classes of phenomena. The information used for the equivalence judges has to be provided by the human analyst for each specific domain based on the divergent equivalents found in the corpus of the final translations of the source text.

5. Evaluation

For calculating precision, we used the whole corpus and all hits of our algorithm. For calculating recall, we manually detected the first two and the last two non-overlapping replacements in each text series files. When less than four replacements occurred in a text series file, we counted each once. There are 55 instances of replacements in our test sample, resulting in a granularity of 2%.[6]

We ran the software over 15 text series files. All but the last were successfully processed (see Footnote 1, Section 2). Out of the 14 remaining files, the replacement detection had 405 hits with 92% precision and 22% recall without any tuning.

A large fraction of the non-recognised replacements were changes in the way things are classified. In a random sample, these accounted for 22% of the cases. To increase recall, we created interchangeability groups that are relevant for the field of activity of our texts, based on the vocabulary of the target text. These groups included representing the same portion of matter either as a 'crumpled sheet' (*zerknülltes Blatt*, *zerknittertes Blatt*, and *verknittertes Blatt*) or as a 'ball' (*Kugel*, *Ball*, and *Bündel*). When we added such thing class interchangeability groups to our German linguistic resource, we increased hits by 113 to a total of 518. Recall rose in the test sample to 25%. Precision rose collaterally from 92% to 94% because of the newly recognised replacements.

For improving recall, we implemented a filter that relies on a user-specified blacklist of classes of things that we do not expect in the current translation task. We manually tagged the wrongly recognised replacements in the list. All of them consisted of segments of words such as *ich* in *Gewicht*, *sich* and *nicht*, as *du* in *durch*, such as *menge* in *zusammengedrückt* and such as *änderung* in *Veränderung*. By adding references to people, sets, types, and the noun *änderung* (changes) to our blacklist, we excluded all and only the 32 wrong hits, resulting in an increase in precision from 94% to 100% without affecting recall.

Inspecting the test sample shows that 31% of the replacements are neither replacements of identical classes nor replacements of interchangeable classes and are therefore not covered by the equivalence judges implemented so far.

6. Discussion

Our method of recovering replacements in text series files has a satisfactory precision and recall for the researched register in our German corpus. Recall can be improved with a better lexical coverage and tuning. The remaining 31% of replacements will be more costly to detect automatically because they involve a variety of linguistic phenomena. These phenomena include:

1. typing of a word spelled in the source language but capitalised according to German graphology: e.g. *Energy* replaced by *Energie*;
2. replacing an elliptic reference by a non-elliptic one: e.g. *[...] Kanten [...] und kleinere werden gebildet* replaced by *[...] Kanten [...] und kleinere Kanten werden gebildet*; and
3. grammatical 'unpacking' of semantic content.

There are two examples of unpacking from phrasal to clause-level units in our test sample: *die in die Speicherung von Energie involviert sind* ('which are involved in the storage of energy') being replaced by *in denen Energie gespeichert wird* ('in which energy is stored') or *Die Erklärung* ('the explanation') being replaced by *zu erklären* ('to explain') (Serbina et al., 2015). Unpackings account for approximately 4% of all replacements. Since such unpackings

[6] Each replacement that is recalled corresponds to an approximate increase of 2% to the total count.

are of particular interest from the point of view of translation studies, this is an aspect that will be addressed in future work.

Two important points remain to be made. First, we noticed that our hit list contains replacements that were hard to detect – e.g. instances when local changes have implications for larger structures – and thus overlooked during manual creation of the test sample. We take this as evidence for the unreliability of manual inspection of such data. Second, the high cost of manual analysis prohibits attempts to increase support for low-frequency phenomena without automation. For this data, we foresee that further replacement types can be detected through the implementation of additional equivalence judges.

7. Conclusion

In this paper, we presented an open-source toolkit for generating text series files, parsing without word boundaries, and detecting replacements in text series files. With this toolkit, we are now able to describe the behaviour of translators with a corpus size and a reliability that was up to now unfeasible with human analysts. We hope to have shown that the effort of providing domain-specific information for the equivalence judges is justified because it gives access to modifications made during the translation process which are otherwise simply not reliably accessible to the human analyst.

In our own research, the tool will be used to analyse which grammatical features are particularly likely to attract attention during the translation process in the form of (repeated) replacements across participants. Moreover, such observations and inferences can contribute to our general understanding of language as a meaning-making resource and to our understanding of why translations tend to display untypical grammatical and semantic patterns.

These tools are not only important for translation studies, but also for research in other areas such as authoring tools (Rösener, 2010) and MT post-editing (Koehn, 2009). Moreover, a parser that does not rely on space boundaries may be used to process spaceless fragments of a text such as Twitter hashtags (Couto-Vale and Hansen-Ampah, 2016), texts in agglutinative languages such as Turkish, and Finnish, and texts in predominantly spaceless languages such as Chinese and Japanese.

8. Acknowledgements

The research reported here was funded by the German Research Council, grant no. NE 1822/2-1.

9. Bibliographical References

Alves, F. and Couto-Vale, D. (2011). On drafting and revision in translation: a corpus linguistics oriented analysis of translation process data. *Translation: Corpora, Computation, Cognition*, 1(1):105–122.

Bateman, J. A., Hois, J., Ross, R., and Tenbrink, T. (2010). A linguistic ontology of space for natural language processing. *Artificial Intelligence*, 174:1027–1071.

Carl, M. (2009). Triangulating product and process data: quantifying alignment units with keystroke data. In Inger M. Mees, et al., editors, *Methodology, Technology and Innovation in Translation Process Research*, number 38 in Copenhagen Studies in Language, pages 225–247. Samfunds Litteratur, Frederiksberg.

Carl, M. (2012). Translog-II: a Program for Recording User Activity Data for Empirical Reading and Writing Research. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association.

Couto-Vale, D. and Hansen-Ampah, A. (2016). The meaning of hashtags matters: detecting hashtags such as #jesuiskouachi. In *Proceedings of the 8th International Corpus Linguistics Conference (CILC 2016)*, page 29, Málaga.

Čulo, O., Hansen-Schirra, S., Neumann, S., and Vela, M. (2008). Empirical Studies on Language Contrast Using the English-German Comparable and Parallel CroCo Corpus. In *Proceedings of LREC 2008*, pages 47–51, Marrakesh.

Cyrus, L. (2006). Building a resource for studying translation shifts. In *Proceedings of LREC 2006*, pages 1240–1245, Genoa, Italy.

Hansen-Schirra, S., Neumann, S., and Steiner, E. (2012). *Cross-linguistic Corpora for the Study of Translations - Insights from the language pair English-German*. de Gruyter Mouton, Berlin.

Helmreich, S., Farwell, D., Dorr, B., Habash, N., Levin, L., Mitamura, T., Reeder, F., Miller, K., Hovy, E., Rambow, O., and Siddharthan, A. (2004). Interlingual Annotation of Multilingual Text Corpora. In Adam Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 55–62, Boston, Massachusetts, USA, May. Association for Computational Linguistics.

Jakobsen, A. L. and Schou, L. (1999). Translog documentation. In Gyde Hansen, editor, *Probing the process in translation: Methods and results*, pages 9–20. Samfunds Litteratur, Frederiksberg.

Koehn, P. (2009). A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Lüdeling, A. (2008). Mehrdeutigkeiten und Kategorisierung: Probleme bei der Annotation von Lernerkorpora. In Maik Walter et al., editors, *Fortgeschrittene Lernervarietäten: Korpuslinguistik und Zweitspracherwerbsforschung*, pages 119–140. Niemeyer, Tübingen.

Rösener, C. (2010). Computational linguistics in the translator's workflow—combining authoring tools and translation memory systems. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing: Writing Processes and Authoring Aids*, pages 1–6, Los Angeles.

Serbina, T., Niemietz, P., Fricke, M., Meisen, P., and Neumann, S. (2015). Part of speech annotation of intermediate versions in the keystroke logged translation corpus. In *Proceedings of the 9th Linguistic Annotation Workshop*, pages 102–111, Denver.

Steedman, M. and Baldridge, J. (2011). Combinatory Categorial Grammar. In Robert Borsley et al., editors, *Non-Transformational Syntax*, pages 181–224. Oxford UK.

Vinay, J.-P. and Darbelnet, J. (1958). *Stylistique Comparée*

du Français et de l'Anglais: Méthode de Traduction. Didier, Paris.

Volansky, V., Ordan, N., and Wintner, S. (2015). On the features of translationese. *Digital Scholarship Humanities*, 30(1):98–118.