Machine Translation Infrastructure and Post-editing Performance at Autodesk

Ventsislav Zhechev
Autodesk Development Sàrl
Rue de Puits-Godet 6
2000 Neuchâtel, Switzerland
ventsislav.zhechev@autodesk.com

Abstract

In this paper, we present the Moses-based infrastructure we developed and use as a productivity tool for the localisation of software documentation and user interface (UI) strings at Autodesk into twelve languages. We describe the adjustments we have made to the machine translation (MT) training workflow to suit our needs and environment, our server environment and the MT Info Service that handles all translation requests and allows the integration of MT in our various localisation systems. We also present the results of our latest post-editing productivity test, where we measured the productivity gain for translators post-editing MT output versus translating from scratch. Our analysis of the data indicates the presence of a strong correlation between the amount of editing applied to the raw MT output by the translators and their productivity gain. In addition, within the last calendar year our system has processed over thirteen million tokens of documentation content of which we have a record of the performed post-editing. This has allowed us to evaluate the performance of our MT engines for the different languages across our product portfolio, as well as spotlight potential issues with MT in the localisation process.

1 Introduction

Autodesk is a company with a very broad range of software products that are distributed worldwide. The high-quality localisation of these products is a major part of our commitment to a great user experience for all our clients. The translation of software documentation and UI strings plays a central role in our localisation process and we need to provide a fast turnaround of very large volumes of data. To accomplish this, we use an array of tools —from document– and localisation–management systems to machine translation.

In this paper, we focus on the effect of the integration of MT in our localisation workflows. We start in Section 2 with an in-depth look at our MT infrastructure. Section 3 focuses on the productivity test we organised to evaluate the potential benefit of our MT engines to translators. In Section 4, we turn to the analysis of our production post-editing data from the last calendar twelve months. Finally, we conclude in Section 5.

2 MT Infrastructure at Autodesk

In this section, we present the MT infrastructure that we have built to support the localisation effort at Autodesk. We actively employ MT as a productivity tool and we are constantly improving our toolkit to widen our language coverage and achieve better perceived quality. At the core of this toolkit are the tools developed and distributed with the open-source Moses project (Koehn et al., 2007). Currently, we use MT for translating from US English into twelve languages: Czech, German, Spanish, French, Italian, Japanese, Korean, Polish, Brazilian Portuguese, Russian, Simplified and Traditional Chinese (hereafter, we will use standard short language codes). We are introducing MT for translating into Hungarian as a pilot this year.

2.1 MT Training Workflow

We start with the training of our MT engines.

Training Data

Of course, no training is possible unless sufficient amount of high-quality parallel data is available. In our case, we create the parallel corpora for training by aggregating data from three internal sources. The smallest source by far consists of translation memories (TMs) used for the localisation of marketing materials. The next source are our repositories for translated UI strings. This data contains many short sentences and partial phrases, as well as some strings that contain UI variables and/or UI-specific formatting. The biggest source of parallel data are our main TMs used for the localisation of the software documentation for all our products.

To ensure broader lexical coverage, as well as to reduce the administrative load, we do not divide the parallel data based on product or domain. Instead, we lump all available data for each language together and use them as one single corpus per language. The sizes of the corpora are shown on Chart 1.

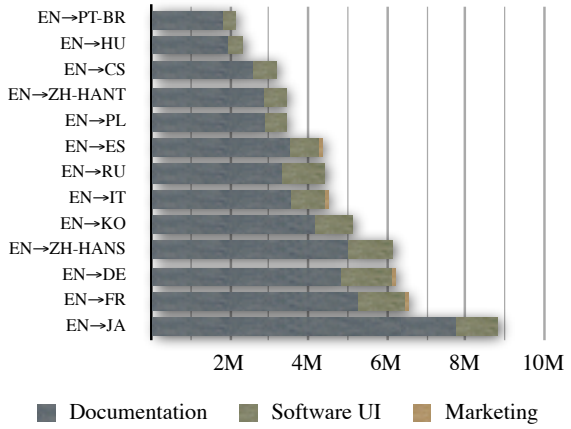

Chart 1: Training Corpora Sizes in Millions of Segments

You may notice that we have the least amount of data for PT-BR and HU, while our biggest corpus by far is for JA. You can refer to this chart when we discuss the evaluation of MT performance — it turns out that language difficulty is a stronger factor there than training data volume.

Data Preprocessing

After we have gathered all available data from the different sources, we are ready to train our MT systems. For this, we have created a dedicated script that handles the complete training workflow. In effect, we simply need to point the script to the corpus for a particular language and — after a certain amount of time — we get a ready-to-deploy MT system. The first step in this training workflow is the preprocessing of the data, which we turn to now.

For the majority of the languages that we support, the preprocessing step consists simply of tokenisation, masking of problematic characters and lowercasing. Some languages require additional preprocessing and we will discuss the details later in this section.

To perform the tokenisation, we have developed a custom Perl tool that consists mostly of a cascade of highly specialised regular expressions. We opted for this tailored approach as our data contains a large number of file paths and URLs, as well as specific formatting conventions and non-content placeholders that could be broken by a non-specialised tool. We also built abbreviation lists based on abbreviations observed in our data.

Another preprocessing step is lowercasing, which is a standard procedure used to improve lexical coverage and reduce lexical ambiguity.

The preprocessing scripts are used both to prepare the corpora for the MT engine training and to handle any data that has been received for translation at run time.

Data Post-processing

Although this is not a part of the training workflow, we will have a quick look at the post-processing tools we use, as they are closely related to the preprocessing tools we just discussed.

Post-processing takes place after a sentence has been translated and we have obtained the translation from the MT engine. As we tokenise and lowercase the data before training, we need to restore the proper case and detokenise the output of the MT engine to make it usable to humans.

For the former task, we use a statistical recaser. This is realised as an additional monolingual MT engine per language which is trained to translate lowercased input into proper-case output. Of course, this adds an additional element of uncertainty and opportunity to produce errors, but with the amount of data that we have available the performance is subjectively reasonable. On the other hand, it is much simpler to maintain statistical re-

casers — they are trained each time we train the regular MT engines — rather than rule-based recaser tools. The latter might require constant adaptation as new data is added to our TMs.

In an effort to recover from some potential errors the statistical recaser might introduce, we have added two specific rules. The first makes sure that the sentence-initial capitalisation of the MT output matches that of the English input. The second rule handles the capitalisation of unknown tokens. These tokens will most likely be new variable names or new URLs that the MT engine does not recognise. The recaser is not able to restore the proper case, which leads to hard-to-detect errors and frustration for the translators. Thus, we make sure that the casing of unknown tokens in the final MT output matches the provided input.

The detokenisation is a much simpler task and is realised as a cascade of regular expressions.

Language-specific Processing

Due to their specific makeup, some languages require extra preprocessing before we are able to handle them with MT. In our case, these languages are JA, KO, ZH-HANS and ZH-HANT.

Firstly, JA, ZH-HANS and ZH-HANT do not use spaces in written text, which makes it impossible to directly use a phrase-based MT system like Moses. We need to segment the data into word-like tokens that will then be used to align against English words. From the available tools on the market, we chose the open-source tool KyTea (Neubig et al., 2011), because it allows us to handle all three languages in question with the same process.

As expected, after translation we need to reverse these preprocessing actions to produce the final MT output. The de-segmentation for ZH-HANS and ZH-HANT is straightforward. We need to take extra care when desegmenting JA, however, as there are cases where the spaces need to remain in place — mostly within transliterated multipart English terms.

A harder issue to resolve with JA arises from the significant difference in syntactic structure between EN and JA, namely, EN is a Subject-Verb-Object language, while JA is a Subject-Object-Verb language. Hence, the linear distance between the verb in the EN source and its translation in the JA target may be very big making it difficult to handle by a phrase-based system like Moses.

Our solution to the problem is to reorder the EN source to make it more Japanese-like, thus reducing the linear distance between corresponding tokens in the EN and JA sentences. First, the EN source is assigned its phrase-based syntactic structure using the OpenNLP parser (`opennlp.apache.org`). Then, we use a rule-based tool developed in-house to move the syntactic heads of the EN sentence to positions corresponding to JA syntax. Our tests have shown this reordering to significantly increase the translators' post-editing productivity, compared to translating from scratch. In fact, using a plain (non-reordered) JA engine does not lead to a meaningful productivity increase, even though we have by far the largest amount of parallel data for the pair EN→JA compared to our other corpora.

Improvements to the Moses Training Toolkit

As stated above, we use the *de facto* standard Moses toolkit for training and decoding. However, early in the process of integrating MT in our localisation workflow, we ran into resource issues during the MT training. The main problem for us was that we could not reliably predict the amount of free disk space that might be required during training, which lead to many interrupted trainings due to our servers running out of disk space. Also, the training process appeared to perform an excessive amount of disk input-output (I/O) operations, which lead to significant slowdowns exacerbated by the particular server architecture we use at our company.

These issues lead us to embark on an initiative to improve the Moses training toolkit to reduce the number of I/O operations and the peak disk usage. As a starting point we took a Moses release from mid-2010, as we considered it the most stable at the time.

The improvements we introduced were focused mostly on avoiding the generation of temporary files during the training process unless absolutely necessary. Where two tools could not directly talk to one another, we used UNIX-style named pipes to handle the data flow, which significantly reduced peak disk usage.

Finally, we noticed that a number of the training steps are independent of one another and could be run in parallel. We exploited this feature by modifying the training script (`train-model.perl`) to run the relevant training steps in parallel. The resulting memory-based data flow during the parallel execution of training steps is shown in Figure 1.

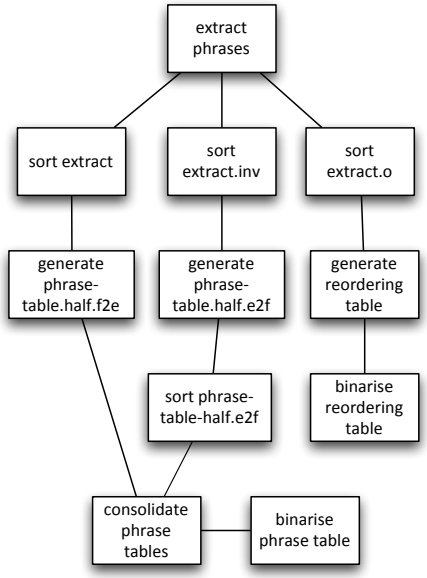

Figure 1: Data Flow for the Parallel Steps of the Moses Training Workflow

A comparison of the peak disk usage and I/O operations during the training of an EN→JA engine with the original and improved workflows is shown in Table 1.

	Original Workflow	Improved Workflow
`extract` file size	7,5GB	uses pipe
`phrase-table.half` size	1,7GB	uses pipe
`phrase-table` size	2GB	uses pipe
`reordering-table` size	2,5GB	uses pipe
total disk I/O	196GB	23GB
peak disk usage	45GB	12GB
disk usage after training	9GB	6GB

Table 1: Disk Usage Statistics for EN→JA MT Training

The modifications to the Moses training toolkit listed above were provided to the MosesCore FP7 project for merging with the main Moses tree.

2.2 MT Info Service

We now turn to the MT Info Service that is the centrepiece of our MT infrastructure, handling all MT requests from within Autodesk. This service and all its components are entirely Perl-based and interact both internally and externally over TCP.

The first element of this infrastructure are the MT servers that provide the interface to the available MT engines running in a data centre. At launch time, the server code initiates the Moses decoder for the requested language, together with any necessary pre- and post-processing tools. The MT servers read data one segment per line and output translations as soon as they are available, with the communication occurring over TCP. For each language that we use in production, we currently have seven MT engines running simultaneously on different servers to provide higher overall throughput.

The MT Info Service itself acts as a central dispatcher and hides the details of the MT servers' setup, number and location from the clients. It is the single entry point for any MT-related queries, be it requests for translation, for information on the server setup or administrative functions. It has real-time data on the availability of MT servers for all supported languages and performs load balancing for all incoming translation requests to best utilise the available resources. In real-life production, we often see up to twenty concurrent requests for translation that need to be handled by the system—some of them for translation into the same language. We have devised a simple and ease-to-use API for communication with the MT Info Service clients.

During the last twelve months, the MT Info Service received over 180 000 translation requests that were split into almost 700 000 jobs for load balancing. Among these requests were over one million documentation segments, as well as a large volume of UI strings.

2.3 Integrating MT in the Localisation Workflow

Once we have our MT infrastructure in place and we have trained all MT engines, we need to make this service available within our localisation workflow so that raw data is machine translated and the output reaches the translators in due course. We use two main localisation tools—SDL Passolo for UI localisation and SDL WorldServer for documentation localisation.

Unfortunately, the current version of Passolo that we use does not provide good integration with MT and requires a number of manual steps. First, the data needs to be exported into 'Passolo bundles'. These are then processed with in-house Py-

thon scripts that send any data that has not been matched against previous translations to the MT info service. The processed bundles are then passed on to the translators for post-editing. Due to limitations of Passolo, the MT output is not visibly marked as such and Passolo has no way to distinguish it from human-produced data. We expect this to be addressed in an upcoming version of the tool.

It is much easier to integrate MT within WorldServer. As this is a Java-based tool, it allows us to build Java-based plugins that provide additional functionality. In particular, we have developed an MT adapter for WorldServer that communicates directly with the MT Info Service over TCP and sends all appropriate segments for machine translation. The MT output is then clearly marked for the convenience of the translators both in the on-line workbench provided by WorldServer and in the files used to transfer data from WorldServer to standalone desktop CAT tools.

WorldServer, however, does present us with its own specific issues to handle — with its use of placeholders (PHs) to mask XML tags. The majority of our software documentation is authored using DITA-based XML and one goal of WorldServer is to hide the XML tags from the translators, as they do not represent actual content. The first issue here is that WorldServer only stores the PHs in the TMs and not the actual content they mask. For example, the segment

The new features of AutoCAD <ver/> are:

will be stored as

The {1}new{2} features of AutoCAD {3} are:

Please note, that any PH may be either an opening or closing formatting tag, or a standalone tag with or without semantic meaning in the structure of the sentence.

An major issue is that in the TMs the PHs are stored with IDs numbered by segment, i.e. in each segment the PHs start from 1; while during translation, the PHs are numbered continuously for the whole project, sometimes reaching IDs into the thousands. This means that any PH with an ID above about 40 will be treated as an unknown word, thus adding significant penalty during translation. We avoid this issue by temporarily renumbering PHs during translation making sure that — for any segment that the MT engines see — the PHs start with ID 1. The original IDs are then restored in the MT output. We found out that, with this process, our MT engines produce very little errors in the placement of PHs and we do not expect to achieve better performance by, say, first removing the PHs and then using word and/or phrase alignment information to reinsert them in the target.

Finally, as most PHs mask formatting XML tags, the whitespace surrounding the PHs is significant. It, however, gets lost during tokenisation and could lead to errors that are hard to identify and fix for the translators. For this, we added an extra processing layer during MT that preserves to the largest extent possible the whitespace surrounding the PHs in the source, regardless of the output of the MT engine and detokeniser.

So far we perused in detail the complex MT infrastructure at Autodesk. The question that arises is if there is any practical benefit of the use of MT for localisation and how do we measure this potential benefit. We present our answer in the next section.

3 Post-editing Productivity Test

We now turn to the setup of our last productivity test and analyse the data that we collected. The main purpose of the productivity test was to measure the productivity increase (or decrease) when translators are presented with raw MT output for post-editing, rather than translating from scratch.

This is already the third productivity test that Autodesk performs. The results of the first test in 2009 are discussed in (Plitt and Masselot, 2010). Each of the tests has had a specific practical goal in mind. With the first productivity test we simply needed a clear indicator that would help us decide whether to use MT in production or not and it only included DE, ES, FR and IT. The second test focused on a different set of languages, for which we planned to introduce MT into production, like RU and ZH-HANS.

The goal of the productivity test described in this paper was mainly to confirm our findings from the previous tests, as well as to help us pick among several MT options for some languages, as well as compare MT performance across products. In the following discussion we will only concentrate on the overall outcome of the productivity test and on our analysis of the post-editing performance versus automatic edit-distance-based indicators.

3.1 Test Setup

The main challenge for the setup of the productivity test is the data preparation. It is obviously not possible for the same translator to first translate a text from scratch and then post-edit an MT version without any bias—the second time around the text will be too familiar and this will skew the productivity evaluation. Instead, we need to prepare data sets that are similar enough, but not exactly the same, while at the same time taking into account that the translators cannot translate as much text from scratch as they can post-edit—as our experience from previous productivity tests has shown. This is further exacerbated by the fact that we need to find data that has not been processed yet during the production cycle and has not yet been included in the training data for the MT engines.

We put together test sets with data from four different products, but most translators only managed to process meaningful amounts of data from two products, as they ran out of time due to various reasons (connectivity issues; picked the wrong data set; etc.). These included three tutorials for AutoCAD users and a users manual for PhysX (a plug-in for 3ds Max).

Due to resource restrictions, we only tested nine out of the twelve production languages: DE, ES, FR, IT, JA, KO, PL, PT-BR and ZH-HANS. For each language, we engaged four translators—one each from our usual localisation vendors—for two business days, i.e. sixteen hours. We let our vendors select the translators as per their usual process.

The translators used a purpose-built online post-editing workbench that we developed in-house. While this workbench lacked a number of features common in traditional CAT tools (like e.g. TM and terminology search), it allowed us to calculate the time the translators took to look at and translate / post-edit each individual segment. For future productivity tests we plan to move away from this tool and use a modified version of Pootle (`translate.sourceforge.net`) instead, as it is easier to manage and provides typical CAT functionality.

3.2 Evaluating Productivity

After gathering the raw productivity data, we automatically removed any outlier segments, for which the translators took unreasonably long time to translate or post-edit. From the remaining data, we averaged the productivity (measured in words per eight-hour business day—WPD) for translating from scratch, taking a specific average for each translator and product combination. We had to use these separate baselines, as the variation between individual translators, as well as between different products for the same translator, is very big.

Comparing to the thus established corresponding baselines, we calculated the apparent productivity delta for each segment that the translators post-edited. The calculated average productivity increase per language is presented in Chart 2.

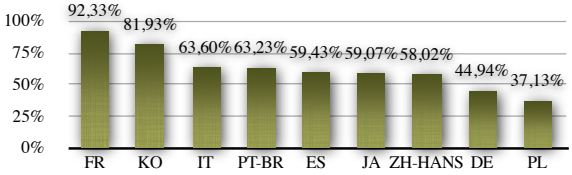

Chart 2: Average Productivity Increase per Language

A caveat is in order here. We need to point out that—due to the setup of our online workbench—we exclude certain translator tasks that are independent of the quality of MT from the productivity calculation. This includes in particular the time that translators would usually spend looking up terminology and consulting the relevant style guides. The calculation also does not include any pauses taken for rest, coffee, etc.

3.3 Analysing the Post-editing Performance

Going deeper, we went on to analyse the post-edited data using a battery of metrics. The metric scores were computed on a per-segment basis so that we could look for a correlation between the amount of post-editing undertaken by the translators and their productivity increase.

The metrics we used were the following. METEOR (Banerjee and Lavie, 2005) treating punctuation as regular tokens, GTM (Turian et al., 2003) with exponent set to three, TER (Snover et al., 2006), PER (Position-independent Error Rate—Tillmann et al., 1997) calculated as the inverse of the token-based F-measure, SCFS (Character-based Fuzzy Score, taking whitespace into account), CFS (Character-based Fuzzy Score, on tokenised data), WFS (Word-based Fuzzy Score). The Fuzzy Scores are calculated as the inverse of the Levenshtein edit distance (Levenshtein, 1965) weighted by the token or character

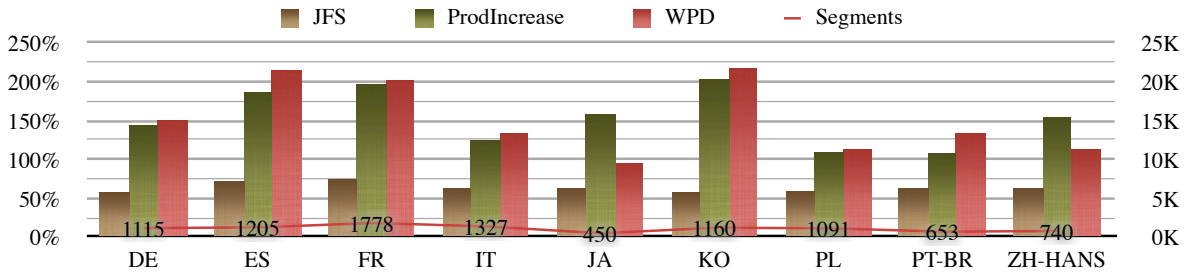

Chart 3: Edit Distance and Productivity Data for All Languages

count of the longer segment. They produce similar, but not equal, results to the Fuzzy Match scores familiar from the standard CAT tools. All score calculations took character case into account.

After calculating the scores for all relevant segments, we obtained an extensive data set that we used to evaluate the correlation between the listed metrics and the measured productivity increase. The correlation calculation was performed for each language individually, as well as lumping the data for all languages together. We used Spearman's ρ (1907) and Kendall's τ (1938) as the correlation measures. The results are shown in Table 2.

	ProdIncrease	
	ρ	τ
JFS	0,609	0,439
SCFS	0,583	0,416
CFS	0,581	0,414
WFS	0,603	0,436
METEOR	0,541	0,386
GTM	0,577	0,406
TER	-0,594	-0,427
PER	-0,578	-0,415
Length	-0,143	-0,097

Table 2: Automatic Metric Correlation with Translator Productivity Increase

We see that among the metrics listed above, WFS exhibits the highest correlation with the measured productivity increase, while METEOR shows the least correlation. The results also show that there is no significant correlation between the productivity increase and the length of the translation. This suggests, for example, that a segment-length-based payment model for MT may not be a fair option. Also, we do not need to impose strong guidelines for segment length to the technical writers.

Considering the results, we decided to look for a possibility to create a joint metric that might exhibit even higher level of correlation. The best available combination turned out to be taking the minimum of SCFS and WFS, which we list in the table as JFS (Joint Fuzzy Score). This metric represents the worst-case editing scenario based on the character and token levels. All other metric combinations we evaluated resulted in lower correlation than WFS. Chart 3 presents the JFS scores per language and the corresponding average productivity increase and post-editing speed. It also lists the total number of segments that were post-edited for each language.

In Charts 4–11, we investigate the distribution of the JFS scores for the different languages tested. The per-segment data is distributed into categories based on the percentile rank. Due to their particular makeup, we separate the segments that received a score of 0% (worst translations) and those that received a score of 100% (perfect translations) from the rest. For each rank, we show the maximum observed JFS (on the right scale). This gives us the maximum JFS up to which the observed average productivity increase is marked by the lower line on the chart (on the left scale). For all languages, we can observe a sharp rise in the productivity increase for the perfect translations, while otherwise the productivity increase grows mostly monotonically.

Additionally, for each percentile rank the left bar on the graph shows the percentage of the total number of tokens, while the right bar shows the percentage of the total number of segments.

We do not include a chart for KO, as it does not appear to follow the monotonicity trend and, indeed, our evaluation of the KO data on its own showed a ρ coefficient of only 0,361. We suspect that this is due to one of the KO translators ignoring the MT suggestions and translating everything from scratch. Because of this peculiarity of the KO data, we excluded it when calculating the overall results shown in Table 1. This also suggests that the productivity increase for KO shown in Chart 2 might not be realistic.

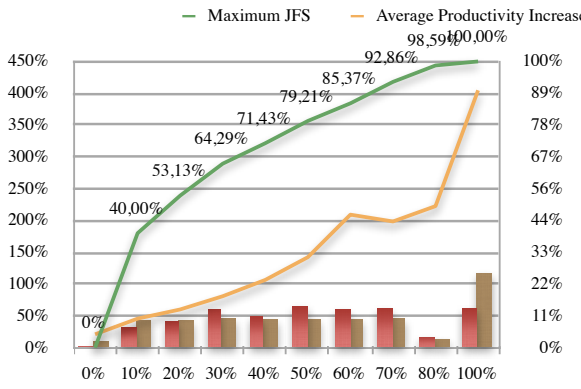

Chart 4: JFS to Productivity Correlation FR

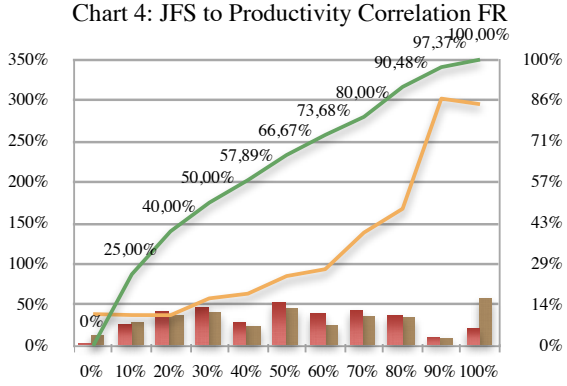

Chart 5: JFS to Productivity Correlation IT

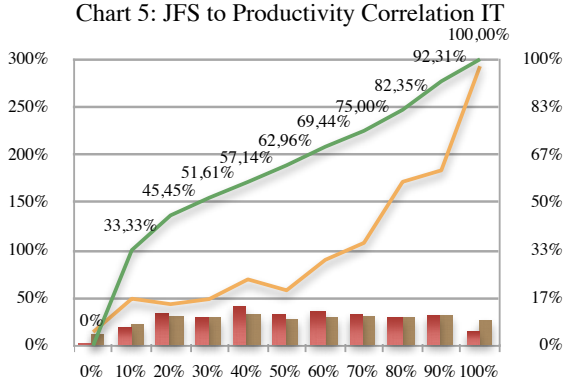

Chart 6: JFS to Productivity Correlation PT-BR

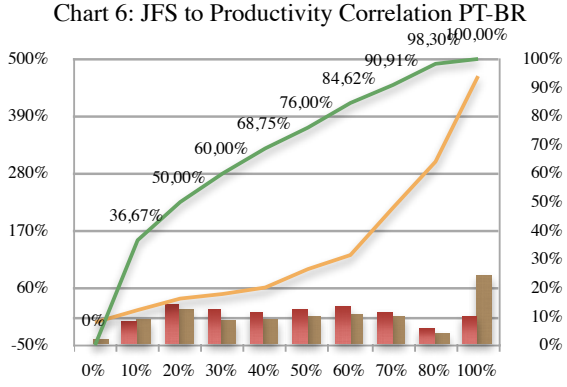

Chart 7: JFS to Productivity Correlation ES

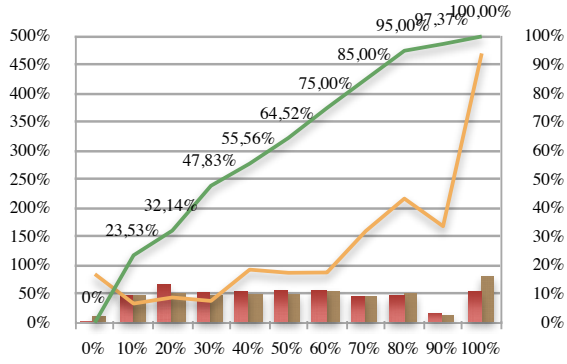

Chart 8: JFS to Productivity Correlation JA

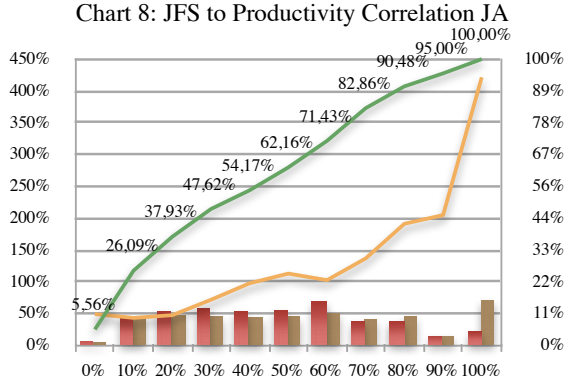

Chart 9: JFS to Productivity Correlation ZH-HANS

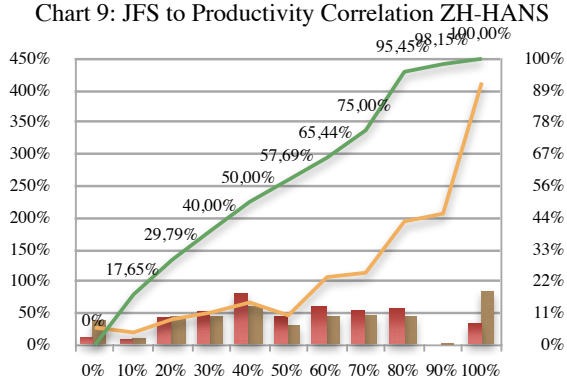

Chart 10: JFS to Productivity Correlation DE

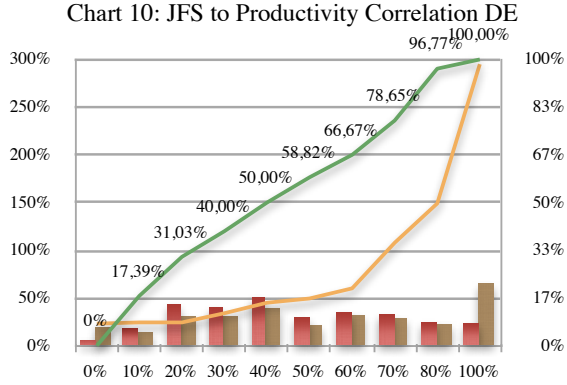

Chart 11: JFS to Productivity Correlation PL

A common observation for all languages is that both the worst and the perfect translations are predominantly short segments, which is as expected. First, it is much easier to achieve a perfect translation for a relatively short segment—especially given that JFS takes whitespace into account and our detokeniser is not perfect. Second, a complete rewrite of the MT suggestion usually results from an out-of-context translation of very short segments.

We also see that the JFS scores for the languages with the highest productivity increase (see Chart 2) are predominantly in the higher ranges, while for DE and PL there is a larger amount of segments with lower JFS.

In the next section, we try to apply the same evaluation methods to real-live post-editing data.

4 Evaluating Real-life Data

A new initiative at Autodesk, which will be extended significantly in the future, provided for the archival of all documentation segments that are post-edited in production. Currently, we store the EN source, the TM or MT target and the final target produced by the translators, but we do not have available any statistics on this data. In the future, we will store the original Fuzzy Match score from our TMs, as well as other metrics that we still need to decide on.

Of course, we do not have productivity data attached to the production segments, as our production environment does not provide for the aggregation of such data. Nonetheless, this is a wealth of post-editing data that we can analyse using the automatic metrics from Section 3.

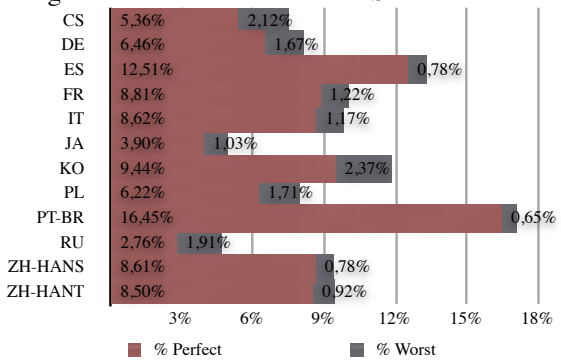

Chart 12: Proportion of Worst and Perfect MT

The first interesting piece of information is the proportion of worst and perfect MT translations, based on the performed post-editing. It is taken as the number of tokens in the worst / perfect translations versus all tokens for each language. Remember that only documentation segments that receive a fuzzy match score below 75% against our TMs are sent to MT. This statistic is presented in Chart 12.

The most important takeaway from this chart is that the proportion of worst translations is negligibly low. On the other hand, there are many perfect translations, despite the disadvantage of Machine Translating only those source segments that were not found in the TMs.

As a further analysis step, we can order the MT engines for the individual languages based on a specific metric per software product. The language order based on the derived JFS metric is presented on Chart 13 for the eight products with the largest translation volume.

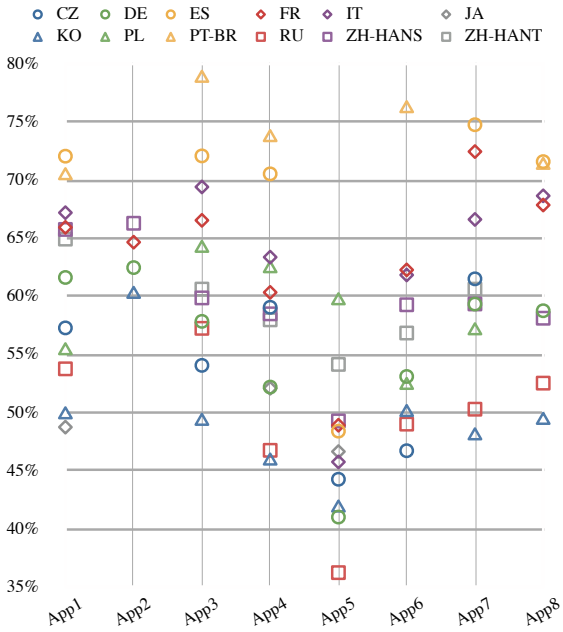

Chart 13: Language Order per Product according to JFS

Although this chart does not include data across all languages for all products, some trends are clearly visible. Namely, ES, IT and PT-BR often present the best JFS, while KO, JA and RU perform poorly on average. While we could expect lower quality MT for KO and JA, the data for RU need an extra explanation. In this case, the poor performance was due to a Unicode-related bug in the recaser for RU that was not detected until late in the production cycle. If we had analysed the data earlier on, we would have found the bug earlier on.

Another trend is for lower performance on average for App5. As it turned out, this was due to one single component within that product, for which the segmentation had failed and many segments contained new line characters. This could not be handled by the MT infrastructure and resulted in MT output that did not match the EN source.

We plan to integrate this type of analysis in a dedicated monitoring system, where we will automatically point our teams to potential issues with the localisation process. This will be accomplished by looking for suspicious patterns in the evolution of the JFS metric — a larger number of over- or under-edited segments may often be to either MT issues or translator under-performance.

For example, we are currently investigating the higher-than-average number of unedited PT-BR segments, given that there we have the smallest training corpus across all languages. We suspect that this could be due to translators leaving the raw MT output unedited without properly checking its correctness. This suspicion is also supported by the presence of a very large number of unedited Fuzzy matches for PT-BR.

5 Conclusion

In this paper, we described the MT infrastructure at Autodesk that is used to facilitate the localisation of software documentation and UI strings from English into twelve languages. We also investigated the data collected during our last post-editing productivity test and found a strong correlation between the edit distance after post-editing and the productivity increase compared to translating from scratch. Finally, we had a look at the post-edited data generated during production in the last twelve months comparing the MT engine performance for some of our products.

We plan to use the insights from the presented data analysis to continuously monitor the performance of our MT engines and for the (semi-) automatic detection of potential issues in the MT process.

References

Banerjee, Satanjeev and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgements. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pp. 65–72. Ann Arbor, MI.

Kendall, Maurice G. 1938. A New Measure of Rank Correlation [June 1938]. *Biometrika*, 30 (1/2): 81–93.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Demo and Poster Sessions of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pp. 177–180. Prague, Czech Republic.

Levenshtein, Vladimir I. 1965. Двоичные коды с исправлением выпадений, вставок и замещений символов (Binary Codes Capable of Correcting Deletions, Insertions, and Reversals). *Доклады Академий Наук СССР*, 163 (4): 845–848. [reprinted in: Soviet Physics Doklady, 10: 707–710.].

Neubig, Graham, Yosuke Nakata and Shinsuke Mori. 2011. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT '11)*. Portland, OR.

Plitt, Mirko and François Masselot. 2010. A Productivity Test of Statistical Machine Translation Post-Editing in a Typical Localisation Context. *The Prague Bulletin of Mathematical Linguistics*, 93: 7–16.

Snover, Matthew, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA '06)*, pp. 223–231. Cambridge, MA.

Spearman, Charles. 1907. Demonstration of Formulæ for True Measurment of Correlation [April 1907]. *The American Journal of Psychology*, 18 (2): 161–169.

Tillmann, Christoph, Stefan Vogel, Hermann Ney, Alex Zubiaga and Hassan Sawaf. 1997. Accelerated DP-Based Search for Statistical Translation. In *Proceedings of the Fifth European Conference on Speech Comunication and Technology (Eurospeech '97)*, pp. 2667–2670. Rhodos, Greece.

Turian, Joseph P., Luke Shen and I. Dan Melamed. 2003. Evaluation of Machine Translation and its Evaluation: Computer Science Department, New York University.