UNIVERSIDADE DE LISBOA
FACULDADE DE LETRAS

A Methodological Approach on the Creation of Trustful Test Suites for Grammar Error Detection

Mestrado em Linguística

Mariana Isabel Pombo Cabeça

2022

Relatório de Estágio especialmente elaborado para a obtenção do grau de Mestre, orientado pela Professora Doutora Helena Gorete Silva Moniz e pela Mestre Marianna Buchicchio

Acknowledgements

Às minhas orientadoras, a Professora Doutora Helena Moniz e a Mestre Marianna Buchicchio, que sempre me apoiaram e mostraram o quão importante é valorizar todo o esforço que fazemos, independentemente dos percalços pelo caminho.

À Professora Helena, por toda a ajuda e apoio desde a primeira aula, mas especialmente por me ter ensinado a ser mais resiliente e segura de mim própria.

À Marianna, pela força de vontade contagiante, pela admiração que cresce a cada dia, por poder tê-la como amiga e, acima de tudo, pelo apoio desde o primeiro dia.

À Unbabel e à equipa de TQT, que me receberam de braços abertos e me permitiram ter a melhor experiência de estágio que poderia pedir.

Aos meus pais que sempre estiveram prontos para me apoiar durante todo o meu percurso académico e por me ensinarem a não desistir e a "descansar em andamento". À minha irmã que tanto me ajudou nas alturas em que mais me sentia bloqueada.

Aos meus amigos, Patrícia, Braga, Catarina e especialmente ao Daniel, por todas as vezes que me encorajaram a dar o meu melhor, que foram sempre um porto seguro para mim e que me ajudaram tão mais do que imaginam.

Index

Abstract

Machine translation's research has been expanding over time and so has the need to automatically detect and correct errors in texts. As such, Unbabel combines machine translation with human editors in post-edition to provide high quality translations. In order to assist post-editors in these tasks, a proprietary error detection tool called Smartcheck was developed by Unbabel to identify errors and suggest corrections.

The state-of-the-art method of identifying translation errors depends on curated annotated texts (associated with error-type categories), which are fed to machine translation systems as their evaluation standard, *i.e.* the test suites to evaluate a system's error detection accuracy. It is commonly assumed that evaluation sets are reliable and representative of the content the systems translate, leading to the assumption that the root problem usually relates to grammar-checking rules. However, the issue may instead lie in the quality of the evaluation set. If so, then the decisions made upon evaluation will possibly even have the opposite effect to the one intended. Thus, it is of utmost importance to have suitable datasets with representative data of the structures needed for each system, the same for Smartcheck.

With this in mind, this dissertation developed and implemented a new methodology on creating reliable and revised test suites to be applied on the evaluation process of MT systems and error detection tools. Using the resulting curated test suites to evaluate proprietary systems and tools to Unbabel, it became possible to trust the conclusions and decisions made from said evaluations. This methodology accomplished robust identification of problematic error types, grammar-checking rules, and language- and/or register-specific issues, therefore allowing production measures to be adopted. With Smartcheck's (now reliable and accurate) correction suggestions and the improvement on post-edition revision, the work presented hereafter led to an improvement on the translation quality provided to customers.

Keywords: Grammar Error Detection; Machine Translation Evaluation; Test Suites; NLP systems evaluation

Resumo

O presente trabalho focou-se na avaliação do desempenho de uma ferramenta proprietária da Unbabel, para detecção automática de erros, baseada em segmentos previamente anotados pela comunidade de anotadores, o *Smartcheck*. Assim, foi proposta uma metodologia para criação de um *corpus* de teste (do inglês *test suites*) baseado em dados de referência com estruturas relevantes (do inglês *gold data*). Deste modo, tornou-se possível melhorar a qualidade das sugestões de correção de erros do *Smartcheck* e, consequentemente, das traduções facultadas. Para além do objetivo inicial, a nova metodologia permitiu assegurar uma avaliação rigorosa, apropriada e fundamentada relativamente às regras usadas pelo *Smartcheck*, para identificar possíveis erros de tradução, assim como avaliar outras ferramentas e sistemas de tradução automática da Unbabel. Recentemente, assistiu-se também a uma fusão da Lingo24 com a Unbabel e, por essa razão, os dados presentes no *corpus* incluem conteúdo traduzido por ambas. Como tal, o trabalho desenvolvido contribuiu inclusivamente para a recente integração da Lingo24.

A *Secção 2* foi dedicada à apresentação da Unbabel, na qual se referem os processos de controlo de qualidade utilizados para assegurar níveis de qualidade exigidos e se descreve pormenorizadamente a ferramenta em foco, o *Smartcheck*. A *Secção 3* focou-se no estado da arte da Tradução Automática e em processos de controlo de qualidade, dando especial atenção a *corpora* de teste e à influência dos mesmos. Além disso, foi também incluída uma descrição relativa ao desenvolvimento de ferramentas automáticas de deteção e correção de erros, criadas para aperfeiçoar os textos provenientes de traduções automáticas.

A metodologia criada, descrita na *Secção 4*, foi dividida em três partes principais: avaliação piloto relativa às regras preexistentes do *Smartcheck*; análise de causas de erros (do inglês *root-cause analysis*); e, por fim, construção de um novo *corpus* de teste, com dados mais recentes e corrigidos.

O primeiro passo na metodologia consistiu na avaliação do desempenho da ferramenta em foco na presente tese. Para tal, foi realizada uma análise piloto na qual cada regra utilizada pelo *Smartcheck* foi avaliada de acordo com métricas comumente aplicadas para avaliação de sistemas de deteção de erros, como o número de verdadeiros positivos (*true positives*) - casos em que o sistema conseguiu corretamente identificar erros -, de falsos negativos (*false negatives*) - casos em que existia um erro, mas o sistema não o identificou - e de falsos positivos (*false*

positives) - casos em que o sistema incorretamente considerou existir erros. Outras métricas utilizadas para avaliação consistiram no cálculo de *Precision*, *Recall*, e *F1-score,* a partir dos valores obtidos das métricas anteriormente mencionadas. Tendo terminado a avaliação piloto, concluiu-se que nem todas as regras foram passíveis de avaliação (razão pela qual se tornou impossível averiguar o desempenho individual para cada regra) e, quanto às que foram avaliadas, os resultados não foram considerados satisfatórios. Isto porque, as regras não identificavam erros existentes nas traduções e consideravam como problemáticos inúmeros segmentos gramaticalmente corretos.

A segunda etapa da metodologia surgiu, então, como tentativa de identificar possíveis razões pelas quais o *Smartcheck* e as regras associadas demonstraram um baixo desempenho. Em vista desse objetivo, foi feita uma análise na qual foi colocada a hipótese de que as regras teriam sido avaliadas com um *corpus* de teste não apropriado e obsoleto, explicando assim as métricas muito baixas da avaliação piloto. Esta hipótese surgiu uma vez que foi não só considerada a possibilidade de os dados do *corpus* não serem representativos das traduções feitas atualmente, mas também pelo facto de as estruturas consideradas problemáticas para os sistemas de tradução serem alteradas constantemente. De modo a corroborar a hipótese colocada, o *corpus* foi analisado com base em variados critérios: qual o tipo de tradução dos dados - se os segmentos analisados tinham ou não sido previamente revisto por pós-editores antes da respetiva submissão; existência de segmentos duplicados ou cujo texto de partida (do inglês *source text*) poderia conter erros - *i.e.* dados ruidosos; e revisão das anotações e das severidades associadas a cada erro, de acordo com tipologias e diretrizes específicas da Unbabel - considerando o número de anotações/severidades correta e incorretamente atribuídas, assim como em falta. Uma vez finalizada a análise, concluímos que cerca de 20% dos dados correspondiam a duplicações - tanto para o registo formal como para o informal -, que entre 15-25% das anotações foram consideradas incorretas e que apenas metade das severidades foram corretamente atribuídas. Assim sendo, considerámos que seria mais vantajoso criar um novo *corpus* representativo e refinado, ao invés de corrigir todas as anotações incorretas do *corpus* previamente usado.

O terceiro e último passo da metodologia consistiu na construção de um novo *corpus* de teste com 27 500 exemplos previamente anotados de traduções automáticas. Os procedimentos para a criação deste novo *corpus* incluíram: filtragem de um conjunto de traduções automáticas, com dados representativos para todas as línguas suportadas pela Unbabel; distinção entre

segmentos dependentes e não dependentes de contexto (uma limitação do *corpus* prévio); exclusão de exemplos duplicados e de casos com textos de partida problemáticos; e, por fim, revisão por parte de linguistas e tradutores das anotações atribuídas, seguindo tipologias proprietárias. Este último procedimento foi ainda subdividido em: uma avaliação geral, de modo a garantir que as traduções transmitiam de forma coerente, fluída e apropriada a mensagem do texto de partida e que, para além disso, seguiam regras específicas para cada língua; uma avaliação focada em especificidades por cliente, de modo a assegurar diretrizes existentes; e uma revisão de severidades associadas a cada anotação.

Tendo sido a metodologia dada como terminada, o *corpus* de teste consistia agora num conjunto de dados de confiança, capaz de avaliar sistemas de tradução automática e ferramentas como o *Smartcheck* de uma forma objetiva e fundamentada. Posto isto, as várias avaliações realizadas - descritas na *Secção 5* - usaram os dados compreendidos no *corpus* como termo de comparação. A primeira avaliação teve como objetivo principal comparar os resultados obtidos na análise piloto quanto às regras do *Smartcheck* com os resultados de uma nova avaliação das mesmas usando o novo *corpus* de teste, de forma a chegar a conclusões mais fiáveis e credíveis. A partir desta, foi possível concluir não só que, contrariamente às conclusões anteriores, todas as regras são agora passíveis de avaliação, mas também que o número de casos em que o *Smartcheck* incorretamente identificava segmentos como problemáticos foi reduzido. A avaliação seguinte comparou anotações recorrendo a uma matriz de confusão (do inglês *confusion matrix*) entre previsões concedidas tanto pelo *Smartcheck* como pelo *corpus* de teste. Deste modo, foi possível identificar quais os tipos de erros mais frequentes e quais os tipos mais (e menos) problemáticos de identificar pelo sistema. Assim, o *corpus* de teste foi considerado como *gold standard* de modo a realizar uma avaliação global do *Smartcheck*, calculando o número total de falsos positivos (atingindo cerca de 45%), falsos negativos (com 35%) e verdadeiros positivos (aproximadamente 20%). Quanto aos verdadeiros positivos, estes foram divididos em dois tipos: segmentos corretamente identificados pelo *Smartc*heck como erro, mas que foram classificados incorretamente (cerca de 11%); e erros em que tanto a extensão como a classificação foram atribuídas corretamente (a rondar os 8% do número total de anotações). A terceira e última análise recorreu aos totais obtidos na avaliação anterior para calcular valores para métricas como *Precision*, *Recall* e *F1-score* para cada língua e para cada registo suportado. Desta forma, foi possível concluir que, quanto à primeira métrica, a média entre registos estava

bastante equilibrada, mas o mesmo não se verificou em *Recall* nem *F1-score*, uma vez que o registo formal atingiu valores superiores. Para além disso, recorremos ainda ao *corpus* para avaliar *spell checkers* usados pela Unbabel e, analisando os resultados obtidos, pudemos concluir que o *spell checker* em uso obteve a avaliação mais baixa. Tendo isto em conta, foi decidido que seria então preferível substituí-lo pelo *spell checker* com a melhor avaliação, de modo a reduzir o número de erros nas traduções e assim melhorar a qualidade das mesmas.

Todo o trabalho realizado pôde ser implementado em vários outros campos para além do inicialmente estabelecido, *i.e.* para além da avaliação sistemática da ferramenta *Smartcheck*. Demonstrando, deste modo, todo o impacto que uma análise bem fundamentada pode ter no processo de tomada de decisão. Isto porque, sem um *corpus* de teste representativo e estruturado, as avaliações feitas não seriam válidas e os resultados obtidos facilmente levariam a conclusões impróprias ou até nocivas para o desenvolvimento dos sistemas e ferramentas em questão.

Palavras-chave: Sistemas de Deteção Automática de Erros; Tradução Automática; *Corpus* de teste; Avaliação de Sistemas de PLN

List of Figures

List of Tables

1. Introduction

This dissertation was written for the Master's degree in Linguistics of the School of Arts and Humanities of the University of Lisbon, in the context of an internship carried out at Unbabel, a Portuguese software company that provides translation services in the Customer Support domain.

The overall interest in the automation of Machine Translation (MT) has been tremendously growing over the years. This is due to the fact that human translation can be considered a demanding and rather complex task, whereas MT provides faster, more efficient and cost effective translations. Nevertheless, MT cannot yet achieve the quality standard that human translation can.

With this in mind, Unbabel successfully combined MT's advantages with the high quality assurance from human revision, thus overcoming a major MT limitation. In order to achieve the best possible translation quality, Unbabel provides translation services that rely on the efficiency of MT outputs and on the resulting quality of subsequently improving them with human post-edition. It is, therefore, of utmost importance to focus not only on the quality of the MT output, but on improving post-editors performance as well.

To that end, Unbabel created Smartcheck, a proprietary grammar error detection tool that highlights possible existing translation errors and provides suggestions in the post-edition stage. This tool was the primary focus of this dissertation and, as such, the work described hereafter aimed to improve Smartcheck's performance and demonstrate the importance of fairness and quality in evaluation data, *i.e.* of creating trustful test suites built upon a robust methodology and gold representative data. Although Smartcheck has several distinct modules, we will focus on the rules and spell checkers assessment.

Evaluation of systems is one of the most crucial steps involved in the MT process. It would not be possible to sensibly assess if the MT outputs accurately conveyed the inputs given without the use of evaluation metrics, which have been improving in parallel with MT development. However, there is one particular aspect that tends to be ignored: we often look into the results obtained from said evaluations without questioning whether or not the conclusions drawn from the assessments are valid and trustworthy. Most likely, this happens due to the fact that we assume that any evaluation standard is created to fit its purpose, to be refined and reliable from the start. Evidently, if the data used to evaluate MT systems is not revised prior to

implementation, it can be extremely harmful to these systems and lead to inaccurate insights and adverse decisions. For instance, a system is being evaluated and it is concluded that a given rule, created to account for a specific translation error, has good coverage and correctly identifies every existing issue related to that same error type in the translation. As such, no adjustments were to be made to that rule. However, in reality, that same rule was classifying various correctly translated words as incorrect. Yet no adjustments will be done, since the rule was evaluated as "good". The opposite can also be the case, where a rule with high accuracy can be incorrectly evaluated as harmful and is later on (unnecessarily) changed or even removed from the system. Additionally, unrevised evaluation data may also contain outdated information or duplicated segments that result in unnecessary longer processing times for the system.

It is of utmost importance to create a sound evaluation standard that contains representative, relevant and accurate data of the content to be translated. Furthermore, the data compiled in evaluation corpora must be consistent in order to avoid multiple classifications for one single segment. For example, if we were to feed a system with two equally translated sentences and consider one as incorrect and the other as correct, then the next time the system encounters that same sentence, it will not be capable of considering it as neither correct nor incorrect, since there is no "correct" decision to be made.

Taking these must-have qualities into consideration, the current dissertation presents a methodology for creating trustful test suites that can be used not only to evaluate different MT systems but also error detection tools, such as Smartcheck. To that end, this dissertation is organized the following way: *Section 2* will describe Unbabel, while focusing on the new and improved quality assurance workflow. Within this workflow, Smartcheck will be exhaustively described as a tool used to assist in post-edition related tasks and, as such, improve the translation's quality. Following this, the state-of-the-art of MT will be presented in *Section 3* alongside a brief history of this field, where the importance and advantages of test suites are highlighted. Additionally, the methodology in this dissertation will be introduced with a baseline analysis (described in *Section 4*) in order to identify possible existing issues, as well as followed by a root-cause analysis. The pilot analysis will afterwards be used as a benchmark to compare Smartcheck's performance prior to and after the new test suites. Lastly, *Section 5* will be dedicated to the discussion regarding the outcome of implementing the methodology described in the previous section. This last section will also include proposals for future related work.

2. Host Characterization

Many platforms have resorted to Artificial Intelligence (AI) to process and translate texts. These AI technologies are based on deep learning, and are stated in the literature as ideal for improving the quality and speed of translation (*e.g.*, Koehn, 2020). Globalization of MT research flourished in the mid 1970s and only recently has a new neural approach emerged, *i.e.* Neural Machine Translation (NMT) (Bahdanau *et al*., 2014; Koehn, 2020). NMT has brought great improvement to the field, particularly on state-of-the-art automatic evaluation metrics (*ibidem*). Taking this into account, Unbabel merges the benefits of AI with the skills of human editors to assemble a hybrid approach of translation.

Founded by Vasco Pedro (CEO), João Graça (CTO), Sofia Pessanha, Bruno Silva, and Hugo Silva in 2013, Unbabel has its headquarters in Lisbon, Berlin, London, San Francisco, and New York. Emerging from values such as creating understanding, embracing diversity, being ambitious and outcome oriented, this translation platform is inevitably prone to thrive. Towards the end of 2021, Unbabel acquired Lingo24 and created a new opportunity to grow in number and diversify its product. Lingo24, with its headquarters in Edinburgh, Scotland, and running a network of five global locations (namely Edinburgh, London, Timișoara, Cebu City, and Panama), has a wealth of experience in delivering high-quality, multi-format, multilingual translations, thus matching perfectly with Unbabel's vision of "building the world's translation layer".

Unbabel MT systems learn from human post-edited data overtime. The combination of MT and post-editing allows for a faster, more efficient, cost effective and high-quality translation, for customer support. With its multilingual supported scale, this Portuguese startup is currently able to manage seventy two different language pairs. Unbabel has also won various awards, such as two ECCCSAs[1] of Best Innovation in Customer Support of 2019 and Best Use of AI and Associated Technologies of 2020, and more recently the CUSTOMER Product of the Year Award of 2021.

In this chapter, we will discuss the following: *Section 2.1* will be dedicated to describing Lingo24's services and its history before the recent integration with Unbabel; in *Section 2.2,* the translation quality assurance workflows will be characterized; in *Section 2.2.1,* we will describe how translation quality is assured through both manual and automated processes. Regarding

[1] **ECCCSAs**: European Contact Center & Customer Service Awards

manual processes, *in Section 2.2.1.1*, Unbabel annotates MT outputs following a specific typology used to both evaluate its community of editors (*Section 2.2.1.1.1*) and MT outputs through annotations (*Section 2.2.1.1.2*). *Section 2.2.1.2* will describe the Quality Framework used, referring to prior services done at Unbabel as well as Lingo24, and what was changed during the integration process. Afterwards, the quality reporting process is explained in *Section 2.2.1.3*. From then on, automatic quality evaluation is described in *Section 2.2.1.4,* referring to different evaluation metrics such as QE and COMET. Finally, the tools used to assure quality and help monitor all processes are introduced and explained in detail. The first tool mentioned in *Section 2.2.2* is the focus of the current dissertation - Smartcheck - and it will be explained how and why this tool is important to editors. Lastly, *Section 2.2.2.1* is solely focused on an interface called Surfboard and its associated rules, as well as how it relates to Smartcheck, by stressing its relation to annotations and, therefore, its relevance on quality assurance processes and workflows.

2.1. Lingo24

An initial disclaimer. Although Lingo24 is now Unbabel and named as such, for clarity purposes, this section will still be describing it as an individual industry. The purpose is to make clearer to our reader what Lingo24 has brought and the challenges that such integration poses, which is also one of the scopes of our work.

Oftentimes, large global companies that generate content at a high velocity find it difficult to scale efforts related to promoting and supporting their products in their native language. This is due to the fact that localization[2] processes are usually highly fragmented and decentralized. By acquiring Lingo24[3] and its services for global enterprises, Unbabel has reinforced its service to better assist global companies, thus accelerating their international growth and engaging their global teams.

[2] **Localization**: According to Reinhard Schäler (2004), "localisation has largely been defined as the linguistic and cultural adaptation of products for specific locales" and therefore "it is the localisation industry that can enable the open, pluralist, user-friendly and inclusive multilingual and cross-cultural information society" (p. 1-2). For further information regarding localization, please refer to *Section 4.3.2*.

[3] **Lingo24 Acquisition**: For more information, please refer to: https://resources.unbabel.com/press-releases/unbabel-acquires-language-translation-company-lingo24

Founded by Christian Arno in 2001, Lingo24 provides translation, localization and consulting services in any language. Due to the fact that Lingo24's 24/7 business model relies heavily on freelancers, it offers the greatest breadth of subject matter expertise in each language pair. This private limited company has also been awarded the Excellence Award for innovative technology at the TAUS (Translation Automation User Society) User Conference in Seattle, in 2012 with the Computer Aided Translation (CAT) tool *Coach*, which will be described in *Section 2.1.1*.

Lingo24's AI-based MT supports translators in around 50% of projects to accelerate turnaround time for customers in all different types of content, without compromising quality. Blending the creativity and expertise of professional translators together with tools to automate repetitive tasks, glossaries and translation memory allows for translations to be tailored to fit the customer's profile and budget. This is done through the AI-powered platform that is built around open source[4] technology components from organizations such as the Okapi[5] Framework, further explained in *Section 2.1.2*.

Additionally, Lingo24 offers a variety of services depending on the customer's needs: different translation services; localization consultancy, achieved through bespoke solutions created to tackle unique challenges and/or LocStrat - *i.e.* localization strategy - assessments; NMT with customized engines to reduce costs and maximize impact tailored to a specific content; international marketing services used to identify keywords in new international markets and optimize foreign language sites; software localization services for a seamless integration, and optimized language assets, checks and quality levels; platform integrations with dedicated translations plugins; and, finally, data simplification used to reduce word count and overall costs, with no impact on translation quality.

Unbabel's AI capabilities and robust multilingual customer service solutions combined with Lingo24's expertise in global enterprises is already emphasizing the ability to generate multilingual content quickly and accurately, by professional translators and proficient bilinguals.

---

[4] **Open source**: According to Deek, F. and McHugh, J. (2007), "the open source movement is a worldwide attempt to promote an open style of software development" where the "products are usually free of direct cost" and anyone is allowed to "modify the code" as pleased.

[5] For more information regarding Okapi, please refer to: https://okapiframework.org/

2.1.1. Coach - a proprietary CAT tool

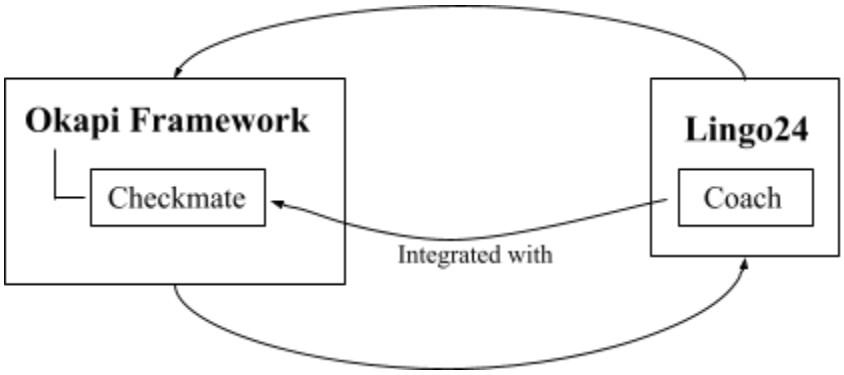
Prior to the description of Coach (Bota *et al.*, 2013) and the Okapi Framework, it is important to note that these tools are presented within Lingo24's chapter and not in the state-of-the-art - *Section 3* - due to their impact within the company and the way they shape the processes at Lingo24. In other words, upon Lingo24's integration into Unbabel, many services were revised so that both companies could compromise on alignment strategies, including the editing assistant tools in question. Moreover, the focus of our work is on distinct tools to assist editors, but, although those tools have similar goals, the way they work and the processes involved are very distinct in nature, due mostly to the two distinct models of translation and the actors involved, *i.e.* Lingo24 mostly relies on professional translators, whereas Unbabel has distinct communities of crowdsourced post-editors, ranging from bilinguals to very experienced professional translators and linguists.

Lingo24 focuses on applied AI to efficiently provide quality and consistency to customers with highly customized engines. With this in mind, a proprietary CAT tool was developed for fastened and accurate translations, called Coach. Coach, compiling customer-specific style guidelines and terminology, is used to automate time-consuming parts of the translation process to deliver faster results and more budget-friendly translations. Thus, not only is the customer able to control the translation's quality, but also personalize quality control checks, regarding, for example, issues with punctuation, spell checking or specific formatting. Moreover, Coach makes it easy for translators to deliver translations with reduced turnaround times. This is due to the fact that there are many customisable interfaces for different tasks. For example, terminology validation, comparative MT revision, live dictionaries to check spelling or synonyms, among others. However, Lingo24 cannot provide such services without the help from a third party, the Okapi Framework. Ergo, Coach is integrated with one of Okapi's utilities API (Application Programming Interface) to perform all the necessary checks.

2.1.2. The Okapi Framework

The Okapi framework is an open source and cross-platform set of components and applications designed to help with localization and translation processes. Okapi is meant to provide tools to build new localization processes or enhance existing ones, while preserving compatibility and interoperability. In other words, Lingo24 is able to take advantage of different

tools that are based on this framework and, as a consequence, take translation's quality one step further. For instance, CheckMate[6] is a commonly known tool that the company relies on to ensure quality. This tool is a graphical user interface (GUI) application with the purpose of performing quality checks on bilingual translated documents. These quality checks are done by looking for repeated words, corrupted characters, patterns in source text that should correspond to a given pattern in the target text, translations suspiciously longer or shorter than the source, and omitted translations, just to name a few operations. One key feature of CheckMate is the possibility of disabling warnings if a specific "error"[7] should not have been detected. This option is important because the next time the verification is re-run, those false warnings, *i.e.* false positives[8], will not be listed again. In addition, this tool allows to integrate the verification that the open-source LanguageTool[9] checker performs, which offers a range of simple and complex checks for many different languages. CheckMate is also backed up by professional linguists and machine learning experts to review MT and improve its overall quality. This utility is of utmost importance, because Coach is integrated with CheckMate's API. In other words, Lingo24's checks are done indirectly through CheckMate, because Coach is the one performing its tasks through this utility, as illustrated in *Figure 1*.



[6] **CheckMate**: https://okapiframework.org/wiki/index.php?title=CheckMate

[7] Note that, in this case, the MT system incorrectly considered a specific token to have an error, *i.e.* a false positive instance. Therefore, the use of quotation marks is solely to facilitate understanding.

[8] **False Positive**: please refer to *Section 2.2.2.1* for a detailed definition.

[9] **LanguageTool**: An open-source project that analyzes the style, tonality and typography of text, aside from giving context-aware suggestions. For more information, please refer to: https://languagetool.org/

Figure 1. Relationship between the Okapi Framework, Lingo24 and main tools

MT outputs can be looked at as a result of a snowball effect in the sense that, if noisy data is fed to MT engines, with no post-edition done, no quality checks performed, and no assessments carried out, then good quality translations cannot be expected as a result. However, if all the steps are improved and scrutinized, then the quality of MT outputs will gradually start to increase overtime. Thus, performing quality checks, having professional linguists and custom engines ready to support any type of content will inevitably lead to better translations.

With this in mind, Unbabel and Lingo24 worked together to try and identify different quality assurance workflows that could have an impact on overall quality and help manage such complex processes.

2.2. Translation Quality Assurance Workflows

According to the Committee for the Coordination of Statistical Activities (CCSA) (2009), "quality is interpreted in a broad sense, encompassing all aspects of how well statistical processes and statistical outputs fulfill key stakeholders' expectations." Every quality aspect must be considered of equal importance, ranging from a sound methodology to data's "relevance, completeness, accuracy, reliability, consistency, timeliness and accessibility" (*ibidem*).

With this in mind, the current section will focus on Unbabel's quality assurance processes such as manual annotations, editors' evaluation, automated metrics, and the quality framework developed upon Lingo24's integration.

2.2.1. Quality Assurance Processes

Unbabel's engines are constantly retrained and improved with post-edited good quality data to ensure consistency and reliability. But how do we know if the data being fed to said systems is good enough to train them?

To guarantee that the data is indeed suitable for its purpose, Unbabel performs distinct quality assurance processes. The goal is to check if the translations' quality is good enough to be sent to the customer. These processes include shared components from the Quality Framework mentioned above and can be divided into two different types: manual quality control and automated processes. The former relates to manual quality assessment methodologies, such as

MQM (Multidimensional Quality Metrics) (Lommel *et al.*, 2014), Editors' Evaluation, and Translation Error Annotations, while the latter concerns Quality Estimation (QE) and COMET (Crosslingual Optimized Metric for Evaluation of Translation) (Rei *et al.,* 2020). Through these metrics, it is possible to determine if a translation is correlated with human judgements and if it is accurate and consistent with the customer's needs.

On one hand, manually evaluating MT can be done through human evaluation. Although this process is considered to be crucial due to its associated intricate analysis and high sensitivity to nuanced errors, it can be expensive and time consuming. An industry standard way of doing so is by annotating errors in a translation and gathering information, in order to use it to generate an MQM score (Lommel *et al.*, 2014), as it will be explained in the following section. On the other hand, evaluating MT can also be done through automatic evaluation when a software is used to generate a score for a given translation. Although this process allows for a lower-cost evaluation, it can never be as accurate as human evaluation nor as sensitive to granular errors. The combination of both manual and automatic quality processes is extensively used at Unbabel for different purposes.

2.2.1.1. Manual Annotations for Quality Assurance

While Lingo24 relied on a variety of quality checks and on DQF-Taus (described in *Section 3.2.1*) to make MQM-based annotations to translated texts, Unbabel adopted two major manual quality processes to ensure quality throughout every translation, namely with Editors Evaluation and Translation Error Annotations through the MQM methodology (Lommel *et al.*, 2014).

When the quality of a translated text is estimated, editors play a major role in the process. This is due to the fact that editors, as the name suggests, edit and correct MT outputs in order for the translated text to be delivered to the client with the highest possible quality. Hence, editors must be evaluated to secure all these requirements, as is described in the following section.

2.2.1.1.1. Communities and their Quality Assessment

Prior to describing Unbabel editors and their hierarchical community, it is relevant to mention evaluators, who belong to the PRO community and that have a major role in the editors'

evaluation. Evaluators are professional translators that work within a proprietary tool called Evaluation Tool by following specific Evaluation Guidelines.

The editor's evaluation is divided into stages as shown in *Figure 2*. The first step is the evaluation of Trainee Editors, through training tasks with fabricated content, while the following stage corresponds to their promotion to become Paid Editors. These ratings allow them to become part of the Editor's Community if their evaluation is positive. They are also allowed to have access to the customers' content once this stage is reached. However, if the evaluation is negative, there are two possible outcomes: if a Trainee Editor receives a negative rating, then he/she remains in the training phase until the following evaluation; if a Paid Editor receives a negative evaluation, then he/she will be demoted to Trainee and carry out training tasks once more. It is worth mentioning that Trainees have multiple opportunities to be evaluated and, if the requirements are met, be promoted. Additionally, editors also follow language and usability guidelines to guarantee a solid foundation for translation.

Figure 2. Unbabel Editor Community Hierarchy

Through the evaluation process, all editors are evaluated on a scale from one to five regarding the quality of their work, as illustrated in *Figure 3*. The further on the scale an editor is rated, the better their translations are.

Figure 3. Unbabel Editor's Quality Scale

Having finished the revision of the MT's output, the resulting translation is delivered to the client and the service is considered completed. Nevertheless, Unbabel takes this as an opportunity to gather these outputs, identify translation errors and categorize them according to the Unbabel Error Typology, an adaptation of the one provided by MQM[10] (Lommel *et al.*, 2014) and developed under Quality Translation 21's (QT21)[11] project.

2.2.1.1.2. Translation Assessment

Unbabel's typology for translation error identification and classification, *i.e.* annotation, was created in compliance with the MQM framework by Lommel *et al.* (2014). The MQM's relevant categories for this dissertation are illustrated in *Figure 4*. The Unbabel's typology version 2 (followed throughout this dissertation) was later replaced by a new version, version 3, created upon Lingo24's integration - mentioned in *Section 2.2.1.2.*

[10] More information about the MQM could previously be found in: http://www.qt21.eu/mqm-definition/definition-2015-12-30.html. For the latest update on the MQM Framework, please refer to: https://themqm.org/.

[11] **Quality Translation 21 (QT21)**: previously in http://www.qt21.eu/. For the latest update, please refer to: https://themqm.org/

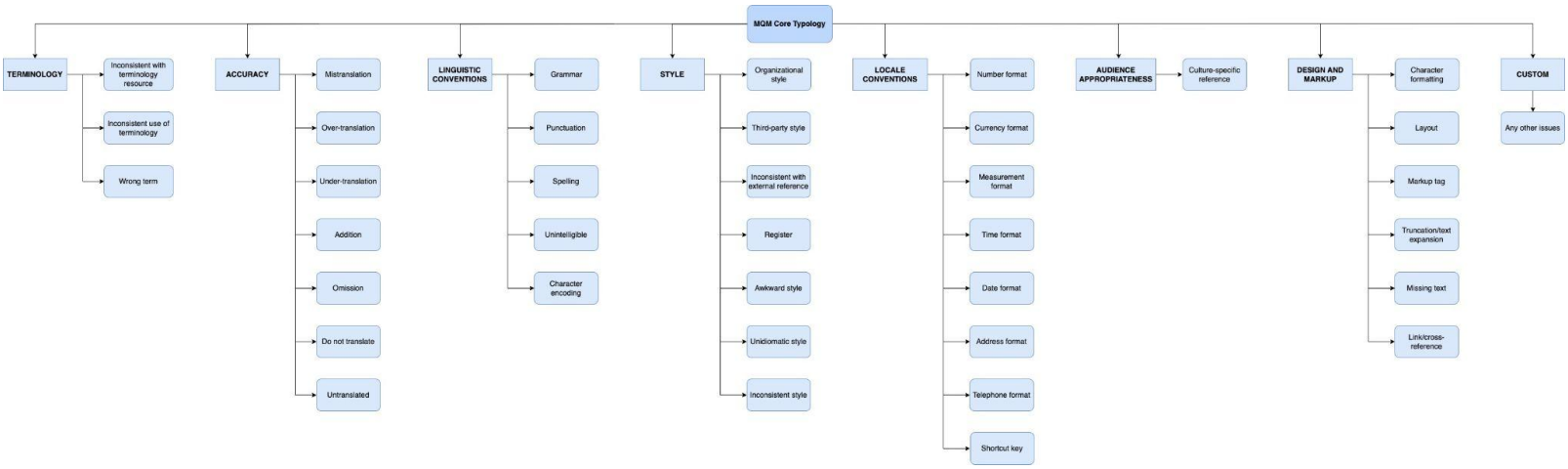


Figure 4. The MQM Core Typology

The annotation process is done by professional translators and linguists, through a proprietary annotation tool called Annotate. This process may be described as the consistent labeling of errors that can fall under three different categories: *Accuracy*, *Fluency*, and *Style*. *Accuracy* errors are linked to the relationship between a source text and a target text, and to which extent the latter accurately renders the meaning of the former. For example, *addition* and *omission* of content are considered to be *Accuracy* errors. On the other hand, *Fluency* errors are related to the linguistic well-formedness of a text (translated or not) and are known to affect readability and comprehension, such as spelling and grammar issues. Finally, the *Style* category includes errors concerning the incorrect use of register or noncompliance with the client's instructions.

Moreover, the degree of severity that an error disrupts a text, and consequently impacts its perceived quality, is associated with one of three different severity levels, namely *Minor*, *Major* and *Critical*. According to the Unbabel Annotation Guidelines, *Minor* errors "do not lead to a loss of meaning", and the overall message being delivered is still understood. However, both the stylistic quality and fluency of the text may decrease. For an error to be considered *Major*, the user's comprehension of said text must be affected. Usually, these errors change an important part of the content, and "the change of meaning results in the improper use of the product/service". Lastly, if an error is offensive, "changes the meaning of the original text and may carry health, safety, legal or financial implications", or if "it violates geopolitical usage

guidelines, damages the company's reputation, causes the application to crash or negatively modifies/misrepresents the functionality of a product or service", then it is considered *Critical*. Note that this severity level requires primary focus due to their impact on translation quality. For this very reason, Unbabel performs targeted tests through Critical Error Test Suites[12] to correct as many impactful errors as possible in a given text.

The MQM score obtained from these annotations, calculated by taking into account the number of words in the target text as well as the number of errors in it, allows us to know how good the quality of a translation is, in that the higher the value, the better the quality. Notwithstanding, this valuable score also allows us to establish an internal process of collecting, assessing and prioritizing translation errors and error types, called the Error Feedback Loop. Thereby, it is possible to holistically analyze the identified errors, report them to teams such as the editor's community and AI teams, in order to continuously ensure higher customer satisfaction and overall increase in quality. Further information about MQM and the scores' calculation, can be found in *Section 3.2.1*.

2.2.1.2. Quality Framework

Unbabel and Lingo24 cooperated to create the current Quality Framework, a new and improved way to report translation quality developed in accordance with MQM scores.

Prior to Lingo24's acquisition, Unbabel offered three different digital Customer Support channels, also referred to as content types. These content types varied according to the translation process and the customer's request. As such, *Chat* (online support messages) solely required MT, due to its need for a shorter turnaround time[13], *Tickets* (a customer support related term used to describe issues raised by customers through emails) required a balanced combination between MT and post-edition, and finally the web facing channel that demanded the

[12] **Test Suites**: Test suites can be considered to be a test set with the purpose of testing the performance of the MT system on specific quality-related aspects. This is done through a subset of test sentences manually chosen that are later given to the MT system. Its performance is then calculated based on the percentage of properly translated instances (Avramidis *et al.*, 2019). For further information, please refer to *Section 3.2.3*.

[13] **Turnaround time**: The time that it takes to fulfill the customer's request, from the moment the customer's order arrives, including the time it takes to translate the order and evaluate its quality, *i.e.* QE, and finally to the moment the client receives the concluded translation.

highest possible quality in translations, *i.e.*, *FAQs* (Frequently Asked Questions). This last content type was considered as having a one-to-many type of communication and, for this reason, required an additional step of human intervention related to translations' proofreading and correction, performed by a Senior Editor[14].

Similarly to the different content types, Lingo24 used to provide different translation service levels falling back on true localization. Ranging from a lower-cost translation to a professional transcreation, there were five different service levels a client could choose from: i) plain MT, a translation provided fully by an MT system with no checking nor editing done by human translators; ii) first draft translation, a light post-edited translation done by a professional translator; iii) professional translation, the one closest to the source and provided with customer specific expertise; iv) on-brand translation, when a professional translator is able to maintain style and tone throughout the text alongside with marketing expertise; and finally v) transcreation, done by a professional transcreator, following a creative writing approach.

Neither the content types mentioned nor the service levels referred to are currently used at Unbabel to determine the translation's quality requested by customers. Due to the fact that it was necessary to meet customer's expectations, Lingo24's integration led to the establishment of a new and revised Quality Framework, evermore focused on fit-for-purpose translation quality. This framework consists of the following major components: restatement of the proprietary MQM-based typology in order to better handle the annotation process, with content from both Unbabel customer support and Lingo24's; the previously mentioned combination between content types and service levels into Quality Levels; the improvement of CUA (mentioned in the following section), as to also be accountable for Lingo24's content; and finally, the setting of Business Critical Errors (BCEs) (Stewart *et al*., 2022).

The Quality Levels created upon the new framework indicate the translation's quality expectation, depending on the expected MQM score. In other words, the higher the quality level requested, the better the translation. Therefore, these levels range from unedited MT outputs all the way to the best possible on-brand translation. Each level is associated with different price-quality relations that allow customers to select the best suited option for them.

Lastly, regarding Business Critical Errors, also referred to as BCEs, they are a subset of errors that indicate which ones can be perceived as *critical* for our customers' needs. In other

[14] **Senior Editor**: A post-editor who consistently provided high quality translations.

words, BCEs go beyond the *Critical Error* definition previously mentioned in *Section 2.2.1.1.2*, as they include errors that are not necessarily considered critical from a linguistic perspective, but that can nonetheless cause damage to the customers.

2.2.1.3. Quality Reporting

Some customers are usually unfamiliar with MQM and its raw subjective scores, and thus understanding what a specific value means might be challenging. Hence, the Customer Utility Analysis (CUA) was created by Unbabel to provide a much straightforward reading of all these values[15]. It is used to report on translation quality and it consists of different quality levels represented by a four color schema, as illustrated in *Figure 55,* with each one being determined by a range of MQM scores, gradually increasing alongside the translation's quality: translations with errors that critically impact the overall communication and meaning are considered to be "Weak" and are associated with a dark red color; translations with discrepancies belong to the "Moderate" category and are associated with a light red color; some translations have grammatical issues that do not interfere with the understanding of the text, and so are considered "Good" translations, with a light green color associated; and lastly, fluent translations with very few minor mistakes are considered to be "Excellent" and are associated with a dark green color.

Figure 5. Example of CUA and its Quality Levels

[15] For more information regarding the process of quality reporting at Unbabel, please refer to: https://help.unbabel.com/hc/en-us/articles/4408078076439-Quality-reporting-Monitor-the-quality-results-for-your-translations and https://resources.unbabel.com/i/1315162-translation-quality-at-unbabel/3?

Therefore, CUA allows customers to have access and a better understanding of these scores and quality measures for the relevant languages, as a consequence of being available on the company's Portal[16]. The Unbabel Portal grants customers the access to these scores and various other metrics, such as information about their requested translations and their particular quality. As such, each customer is able to audit their own data and even filter it to a specific time period, if desired.

2.2.1.4. Automated Quality Metrics for Quality Assurance

Unbabel aims to achieve the highest quality possible for MT, hence the necessity to know how reliable the translation is and how good the MT engine performs. For this reason, Unbabel created its own framework for QE to account for the former concern, and opted to mainly rely on the evaluation metric COMET for the latter.

Although editing MT is usually faster than translating sentences from scratch, some of these translations end up being more complex than expected, due to multiple factors (*e.g.* context related issues, wrong lexical selections, misspelled words) and consequently, their corrections become time-consuming. "Therefore, estimating post-editing effort to support the work of translators is a desirable feature in computer-aided human translation workflows" (Specia *et al*., 2018, p.46). This is exactly what QE does, because it predicts the quality of an MT output. It is trained with both the original sentence from the customer's order and the target sentence, without access to a reference.

With this in mind, Unbabel created its own open source framework called OpenKiwi (Kepler *et al.*, 2019), also known as Kiwi. Kiwi can perform on two separate levels, word-level and sentence-level, and provide a quality score for each one. Word-level QE aims "to assign quality labels (OK or BAD) to each machine translated word, as well as to gaps between words (...), and source words" (Kepler *et al.*, 2019, p.117-118). Demonstrated in *Figure 6* is an example of a training set, in which the top sentence in English is the source, the bottom sentence is the MT output in German, and the middle sentence corresponds to its manual post-edition. In QE, there are three types of word level tags: MT tags that account for words that are replaced or

[16] **The Unbabel Portal**: It ensures that customers have access to valuable insights and particularly to the metrics used to evaluate translation quality within all channel types and language pairs. For further information, please refer to: https://unbabel.com/portal/

deleted; Gap tags to account for words that need to be inserted in the sentence; and Source tags, used to indicate missing or mistranslated source words. This way, the classification is done in a meticulous manner, taking into account issues such as context, mistranslations, and omissions.

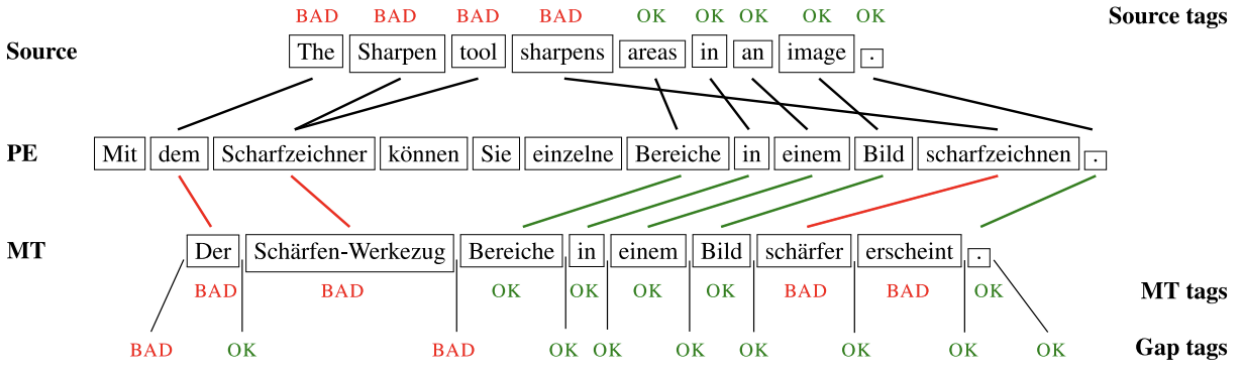


Figure 6. Example of a Word-level QE Training Set (Kepler et al., 2019, p.118)

On the other hand, the goal of sentence-level QE is to predict the translation quality of the entire sentence, focusing on a greater scope. Therefore, the final score is calculated with Sentence HTER and Token Scores associated. Sentence HTER (Snover *et al.*, 2009), also known as Human-Targeted Translation Error Rate, is a quality evaluation metric that predicts how many edits will be necessary to achieve a good quality MT. It ranges from 0 to 1, in that the higher the score the more edits that need to be made in a given sentence. The other value mentioned - Token Scores - relates to the accuracy of translation regarding each individual word, meaning that each word receives a score - a token score - from 0 to 1, and the higher the score the worse was the word choice for that specific translation.

Although we may know how reliable a given translation is, the second concern still remains. How do we ensure that the MT engine is fit for its purpose? “To judge the quality of a machine translation, one measures its closeness to one or more reference human translations

according to a numerical metric" (Papineni *et al.*, 2002, p.311). In other words, the goal is to see if the hypothesis[17]' quality resembles the reference[18]'s, while resorting to automatic metrics.

Currently, Unbabel makes use of various automatic evaluation metrics for MT. Notwithstanding, we will now focus on introducing Unbabel's proprietary metric, COMET (Rei *et al.*, 2020), which is used for leveraging retrainings of MT. COMET is a refined metric based on already existing ones that poorly correlate with human judgments. To surpass this correlation limitation, Unbabel created a machine learning system that is trained with two types of data, one from open source models and another from internal models. Regarding open source models, COMET uses pre-assigned Direct Assessment[19] scores on translation to improve the correlation. Whereas internal models are trained with in-house data, for which we have much more granular MQM annotations and scores. For this reason, COMET's correlations with human judgment are much higher when compared to other metrics, since all data used to train it is directly provided by human translators (Rei *et al.*, 2020). Moreover, COMET's performance regarding correlation between MT output and human judgment is one of the highest, scoring a 0.6 on a scale from 0 to 1. When compared to other metrics such as BLEU, our internal metric also shows higher correlation scores for a vast number of categories, namely addition, capitalization, mistranslation, omission, spelling and word order. However, recent analyses have revealed that COMET still struggles with fine-grained differences, such as named entities, and punctuation-related issues.

Recently, a tool that helps visualization of MT performance was created, called MT-Telescope[20] (Rei *et al.*, 2021). Using this tool, it is possible to compare the output quality of two MT systems by looking for disparities and generally evaluating the distribution of quality scores (*ibidem*).

System's performance evaluation can significantly grow if it takes into consideration multiple other metrics to achieve continuous improvement. This being said, Unbabel still makes

---

[17] **Hypothesis**: A translation of the source text (not necessarily MT, since any translation can be considered an hypothesis).

[18] **Reference**: A possible translation of the source that has been verified by a human translator (*i.e.* a 'gold standard').

[19] **Direct Assessment**: According to Elbeck, M. and Bacon, D. (2015, p.2), direct assessment "involves scoring a learner's task performance in which the performance is believed to be contingent on achieving a learning goal".

[20] For more information about MT-Telescope, please refer to: https://unbabel.com/research/mt-telescope/

use of other types of metrics for benchmarking purposes such as BLEU[21] (Papineni *et al.*, 2002) and METEOR (Lavie and Denkowski, 2009).

2.2.2. Smartcheck - a Proprietary Editing Assistant Tool

As part of the Translation Quality Workflow, alongside the framework just described, Unbabel created an error detection tool called Smartcheck, that identifies existing translation issues to support editors in post-edition by providing them with possible corrections. In other words, this editing assistant tool helps editors to find translation issues much faster and to avoid overlooking other possible faults, focusing exclusively on content derived from *tickets*.

When Lingo24 merged with Unbabel, their error detection processes and translation checks were integrated into the already existing quality assurance tools at Unbabel. Thus, different checks from CheckMate, a tool which belonged to the Lingo24's error detection process, were integrated into Smartcheck. This allowed for all content (both from Unbabel and Lingo24) to be taken into account by this error detection tool.

In this section, Smartcheck's architecture is explained taking into account its relation with external services and its dependency on proprietary grammar checking rules. Moreover, it is described as a tool of great importance to post-editors, due to the support it provides when performing text editing tasks. The current section is focused on how these proprietary rules are created, evaluated and deployed to Smartcheck, allowing editors to make use of them in their daily tasks.

As previously stated, one way to ensure MT quality is through human evaluation, post-edition and translation error annotations through MQM. Due to the constant checking of translations to improve and retrain the engines, editors are essential to Unbabel's success. But how can we help them do their job, spur them to accomplish tasks, reduce errors and assist them while taking into account time limitations? Here is where Smartcheck comes into play and has a crucial role in assisting editors.

Once a customer provides Unbabel the source text to be translated, it can follow through different pipelines depending on the customer needs. However, the process involved in editing

[21] **BLUE (Bilingual Evaluation Understudy)**: An automatic machine translation evaluation method that correlates highly with human judgments by averaging out individual sentence judgment errors over a test corpus (Papineni *et al.*, 2002).

the MT output is always done by Unbabel's Communities assisted by Smartcheck. As illustrated in *Figure 7*, Smartcheck checks for errors such as register and formality inconsistencies, specific client rules, and overall consistency of the text, while the spelling portion is handled by external Natural Language Processing (NLP) services, such as a word aligner, a syntax parser, and a spell checker. The first NLP service mentioned - the word aligner - is responsible for finding the correspondence between source and target words in a pair of sentences. Additionally, the syntax parser named Stanza[22] plays a crucial role when handling morphological issues. Stanza is a natural language analysis package that allows Unbabel to convert text strings into lists of sentences and words, to later generate their morphological features as well as their syntactic dependencies and, as such, recognize named entities. Finally, regarding the spell checking service, Unbabel used to rely on Aspell[23] to replace misspelled words in English and, thus avoid issues that would derive from source texts. However, this tool was recently replaced by Hunspell[24], which can morphologically analyze languages through word-level writing systems with complex compound and character encoding.

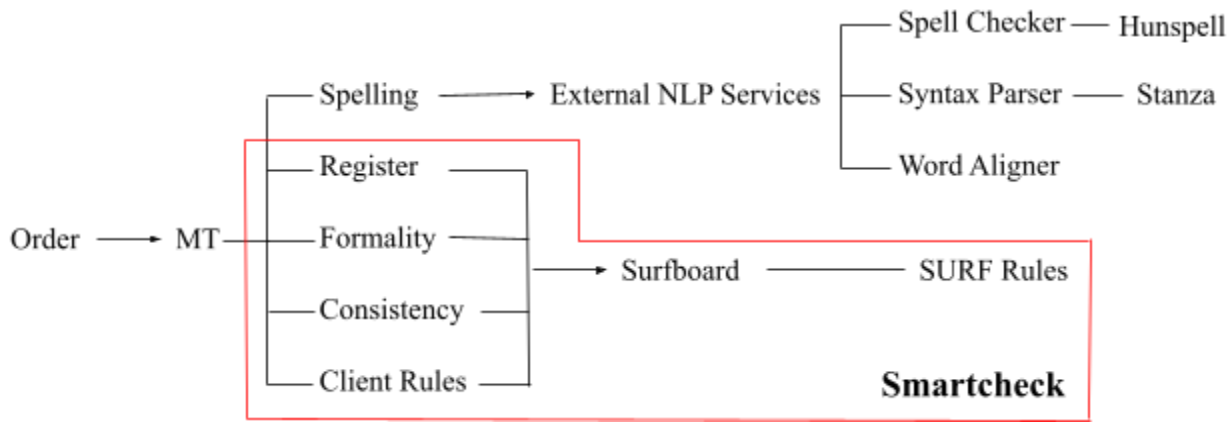


Figure 7. Smartchek's Architecture

Smartcheck helps editors in performing their translation tasks faster and more efficiently, due to less mistakes being overlooked, and in making it possible to decide if a highlight error

[22] For more information regarding Stanza, please refer to: https://stanfordnlp.github.io/stanza/

[23] For more information regarding Aspell, please refer to: http://aspell.net/

[24] For more information regarding Hunspell, please refer to: https://github.com/hunspell/hunspell

needs to be changed. One key feature of Smartcheck is that it encompasses custom language rules written in a proprietary programming language called SURF for various types of issues, such as style, fluency, grammar related errors, dependency problems, among others. Therefore, Smartcheck is the result of a combination of several NLP modules and hardcoded rules for different language pairs.

This tool can be seen as a supercharged multilingual version of a spell checker, as it analyzes not only grammar and orthography, but also morphology and style-adapted client rules, going beyond the spell checkers and morphosyntactic classifiers. Moreover, it identifies possible mistakes and suggests specific corrections. In other words, it does not substitute the mistake for the correct form, and for this reason it must be considered a Grammar Error Detection (GED)[25] tool instead of a Grammar Error Correction (GEC)[26] tool. The core difference is that the editor has the final word and they must decide to accept or reject the suggestions.

Smartcheck's purpose is to help editors in two major ways: by checking potential grammatical errors and by automatically suggesting tips concerning customer's style guides. This tool operates in the Unbabel's Editors Interface and it was developed using proprietary technology based on several language rules that check the translation coherence and consistency. When Smartcheck detects an error on a segment, it underlines that expression in red - illustrated in *Figure 8* -, in order for the editor to check the provided suggestion and subsequently decide whether to edit the suggestion or not. In the example below, the required register is highlighted in the top right corner and thus, the translation must have a formal tone. However, an informal pronoun, *i.e.* the "ti" circled in red, is incorrectly used instead.

[25] **Grammar Error Detection (GED):** According to Rei and Yannakoudakis (2016), GED is independent from GEC as it detects grammatical errors by sequence labeling each token in a sentence with a binary classification (*i.e.* correct or incorrect), and classifies them accordingly, depending on the grammatical category they belongs to. For more information regarding GED, please refer to *Section 3.3.1*.

[26] **Grammar Error Correction (GEC)**: "The task of correcting different kinds of errors in text such as spelling, punctuation, grammatical, and word choice errors", as described in the shared task of GEC in http://nlpprogress.com/english/grammatical_error_correction.html (accessed 5 July 2021)

Figure 8. Example of a Smartcheck's suggestion regarding the incorrect use of register in an MT output from English to Italian

As a result, Smartcheck can help editors in many ways. For example, they are less likely to overlook misspelled words, agreement issues and inconsistencies regarding register. Additionally, they can avoid the time-consuming task of reading numerous customer's style recommendations and associated customer-specific terms. Therefore, it is possible to significantly improve the editor's work quality and change the entire process to be more time-efficient and, consequently, transform this step in the translation pipeline to be regarded as cost-effective.

All things considered, Unbabel created Smartcheck due to the flexibility and increased control it provides when compared to other GED tools. The possibility of creating language-specific rules simply by following an unique syntax, allows for greater control in the MT process and for creating unlimited custom rules for clients. Thus, Smartcheck makes it possible for editors to achieve higher quality in translations and, as a consequence, use these texts to feed MT systems and improve them over time. In short, the more the editors edit MT outputs and correct translation errors, the better the outputs of the systems will become.

2.2.2.1. Surfboard

Smartcheck heavily relies on coded rules from Surfboard, a proprietary interface which allows for rule creation, testing and evaluation. It is important to draw a distinction between Surfboard and SURF, since the former is a back-end platform and the latter a proprietary programming language used in said platform to develop custom rules*, i.e.* SURF rules. These rules are then deployed to Smartcheck to assist editors and improve their performance.

Managing rules can be a challenge, especially when there are distinct rules per language, customer and within customers also per brand. With this in mind, the following question arises: how do we know if the existing SURF rules are accurate enough to fit their purpose? In order to answer that question, each rule's performance must be evaluated. The state-of-the-art process of assessing the rules' performance is through evaluation with test suites. With regular assessments and better test suites to evaluate the custom rules, the systems can be upgraded, improved and even more precise.

When creating a system that automatically translates texts from one language to another, it is extremely important to have a considerable amount of examples in both the source language and the target language corpora. However, human language is not as simple as direct translations and so only relying on corpora for MT is neither efficient, nor realistic. It would be impossible to account for every sentence and structure possible in one or more languages. For instance, expressions that depend on context, cases of ambiguity, and even the fact that some languages have different grammatical features are some examples of this limitation. Thus, instead of creating grammar rules that only account for specific expressions, what is crucial is to create rules that cover a broad range of common translation errors. Moreover, there needs to be client-specific rules to not only account for explicit guidelines, but also for brands within those clients. For this reason, parallel corpora and curated data that account for customer specificities are of utmost importance.

Although editors review the translations, there will inevitably be overlooked errors. Regardless, recognizing the problem, and taking corrective actions by combining both custom grammar rules and quality in extensive data is crucial. For this reason, improving MT engines is a very meticulous process, involving not only the need to assist editors, to know what are the most frequent critical translation errors and to create rules that avoid forthcoming problems, but

also the fact that these systems need to be frequently tested and that the problematic structures must be addressed and corrected.

With this in mind, Surfboard is the tool designed for the creation and regular testing of language rules, by checking their metrics and automatically evaluating their quality. As previously mentioned, the programming language for writing grammar checking rules is called SURF. SURF, a domain-specific language (DSL), has an unique syntax for writing code and also a compiler to turn it into a machine-understandable format. This proprietary programming language has two main advantages: the first one is that custom-made rules allow for an easier and more efficient application in staging - a "pre-production" environment for testing before deployment -; and the second one is the fact that specific language rules can be easily improved and have new features added, if needed, while automatically measuring its performance. This way, the process of creating grammar rules becomes user-friendly and, consequently, it allows for evaluation of every rule in production.

But how does Surfboard actually connect to Smartcheck? In order for SURF rules to properly assist editors, they all need to be gathered and tested on Surfboard in the first place, so that afterwards Smartcheck can retrieve them. To enable running SURF and simultaneously check the source text (input), a Python server had to be created and divided into two standard parts: an interface and an engine, as illustrated in *Figure 9*. The interface is offline and backed by a compiler for supporting the writing of the rules. On the other hand, the engine is online and always running the rules against received text, *i.e.* it stays running and constantly waits for texts to be checked. Once the MT output is provided, Smartcheck will get all the current rules from Surfboard and save them on the SURF Engine, using a "GET" rule. Every time a rule is added, modified or deleted, Surfboard notifies Smartcheck, via a request with the ID of the rule and the action needed - either adding a new rule, updating an existing one, or deleting it, if necessary. It is important to mention that if a rule is updated, then Smartcheck will ask Surfboard for the rule string via a query and then save it on the SURF engine, and if it is deleted, then Smartcheck will delete the rule from the SURF engine itself.

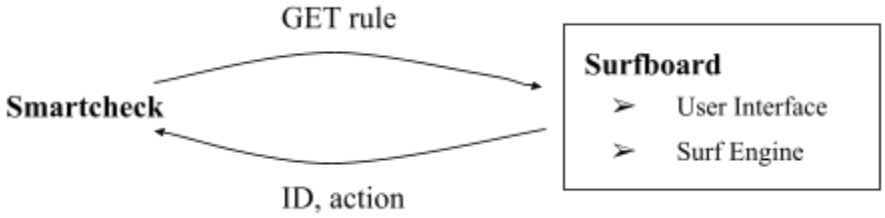


Figure 9. Communication between Smartcheck and Surfboard

Moreover, in Surfboard every rule can be evaluated using a series of different metrics built-in the platform, such as #F to account for how many times a rule was fired, TP for cases of true positives, FP for false positives, FN for false negatives, and lastly #A to account for how many times it was annotated, *i.e.*, obtained by summing cases of True Positives (TPs), False Positives (FPs) and False Negatives (FNs) per rule. While a TP is a case of an error being correctly detected by SURF rules, a FN is a case of an existing error not being detected, since the classifier - SURF - considers a word as not having an error - negative class[27] -, when it should have. Lastly, cases of FPs consist of expressions where errors were detected when there were no actual errors. Thus, they are classified as positive, but the positive/gold class[28] was negative.

Surfboard can also calculate metrics such as *Precision* (P) and *Recall* (R), which are standard and widely used to evaluate systems' performance (Makhoul *et al.*, 1999), in the following way:

$$Precision\ (P)\ = \frac{True\ Positives}{True\ Positives + False\ Positives}\ ,\ Recall\ (R)\ = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

These are often combined with the *F1-score* metric, in case a numerical measurement of a system's performance is required. Therefore, *F1-score* can be defined as "the weighted harmonic mean of *P* and *R*" (*ibidem*, p2):

[27] **Negative Class**: When a word/expression is correctly translated and has no grammatical/lexical related problems, *i.e,* absence of issues.

[28] **Positive/Gold Class**: When a word/expression has grammatical/lexical related problems and is therefore incorrectly translated, *i.e.,* presence of issues.

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

The ideal situation when testing a rule is for it to achieve high metrics and avoid cases of false positives and false negatives. SURF rules rely heavily on MQM annotations for providing the error categories. This is the reason behind the existence of test suites that are used to evaluate every rule in Surfboard. These test suites, combined in one single evaluation dataframe, account for every language pair Unbabel supports and must consist of gold annotations. Since annotations are used to evaluate SURF rules, hardcoded Smartcheck rules, and Unbabel's spell checker they must be curated and customer-representative. As mentioned earlier, knowing which structures are problematic for each language, which grammatical categories need more attention, and what are the overall strengths and weaknesses is a critical step when creating rules to fix translation problems. Hence, the greater the thoroughness of annotations, the better the quality of the MT output. For this reason, the grammar rules created for overlooked annotation errors are established with the annotation's Error Typology created by Unbabel.

3. State of the Art

MT's research has been increasingly expanding over time, despite some hardships along the way. Nonetheless, there is still much to do in order to truly guarantee good quality outputs and the system's full automation.

In this chapter, we will start by discussing the initial attempts at developing MT systems in *Section 3.1*, and describe multiple efforts to improve the systems' performance and overall progress. Over the years, the investment in the MT field led to an exponential growth in these systems and new approaches started to emerge. Approaches such as rule-based, statistical, and more recently NMT. However, with the expansion of MT research, the need to assure quality and evaluate these translation systems also became a necessity. Thus, manual and automated metrics such as MQM (*Section 3.2.1*) and QE, BLEU, METEOR, and COMET (*Section 3.2.2*), were created to enhance quality assessment processes. Additionally, for an evaluation's result to be reliable, the gold standard must compile extensive, representative and revised data, *i.e.* the evaluation must be done using test suites. *Section 3.2.3* is dedicated to describing test suites, their purpose, progress throughout the years, and the different approaches to adopt when creating them. Finally, in an attempt to improve the quality of machine translated outputs, tools of GED and GEC were created. These tools, described in *Section 3.3*, also allow for a decrease in human dependency when reviewing translations and identifying potential existing issues.

3.1. Brief history of Machine Translation

MT came into the spotlight in the early Twentieth Century with electromechanical devices capable of being used as translation dictionaries. In 1933, Georges Artsrouni, a French engineer, created a "Mechanical Brain" (*Cerveau Mécanique*) that functioned as a multilingual dictionary for producing quick rough translations. The main problem with Artsrouni's creation was its underlying inability to provide more accurate translations. That same year in Russia, Petr Petrovič Trojanskij developed a "translating machine" capable of bilingual and multilingual translation. MT research was therefore finally taking its first steps.

In 1949, Warren Weaver wrote a memorandum entitled "Translation" that is said to be the "single most influential publication in the early days of machine translation" (MT News International, 1999, p.5). By then some word-for-word automatic translations started to appear, but they were too crude and consequently this approach presented many limitations. Taking this

into consideration, Weaver's memorandum put forward four new proposals to overcome these limitations: i) problems of ambiguity may be solved through context and, depending on the noun, verb or adjective with multiple meanings, the amount of context required changes; ii) there are logical elements in all languages, so any written language is an expression of logical characters, and thus the problem of translation can be formally solvable; iii) the frequency of letters, their combinations, patterns, and intervals between each may be independent to some degree of the language used; iv) the fourth proposal focused on the premise of linguistic universals, derived from the second proposal's logical features.

Although some rejected these proposals, others saw this as an opportunity to dive into the unknown world of MT. Hence, in the beginning of the 1950s, research started to bloom with Yehoshua Bar-Hillel in Massachusetts Institute of Technology (MIT) due to the growing interest in Weaver's memorandum. For Bar-Hillel, research on MT could "(...) probably provide valuable insights into the functioning of linguistic communication" (Bar-Hillel, 1951, p.229), particularly regarding syntax and how it could deal with translation ambiguities. However, these were but the groundwork in the MT's progress, as fully automatic translations were far from being achieved:

> "... fully automatic translation would not be achieved without long-term basic research, and (in the interim) human assistance was essential, either to prepare texts or to revise the output (known already as pre- and post-editing." (Hutchins, 1995, p. 433).

Soon after, Léon Dostert in collaboration with the International Business Machines (IBM) Corporation did the first public demonstration of an MT system, albeit not yet owning the adequate computer facilities. Nonetheless, this demonstration gave inspiration for other countries to invest in other MT projects and, as a consequence of the Cold War and the rise of MT, the US focused the research on Russian-to-English translations whilst most Soviet research was directed at English-to-Russian systems (Hutchins, 1995).

For a fleeting moment, the assumption that MT was heading in a promising direction was spreading. However, in the beginning of the 1960s it shortly faded into disillusion:

> "Those who are interested in MT as a primarily practical device must realize that full automation of the translation process is incompatible with high quality. There are two possible directions in

> which a compromise could be struck; one could sacrifice quality or one could reduce the self-sufficiency of the machine output." (Bar-Hillel, 1960, p.93)

According to Bar-Hiller (1960), even human translators need extra-linguistic knowledge to provide high quality translations. For this very reason, even ambiguous structures cannot be settled simply through context, as Weaver believed. Thus, Bar-Hiller concluded that fully automatic high quality translation (FAHQT) can only be achieved with the cooperation of human post-editors to fix these flawed MT outputs.

The lack of interest, and consequently of funding, in MT grew significantly in 1964 with a report from the Automatic Language Processing Advisory Committee (ALPAC). This report, considered by some as "narrow, biased, and shortsighted" (Hutchins, 1995, p.436), stated that there should not be further investment in MT research due to its time-consuming nature, the great cost associated, and the low-accuracy outputs. Eventually, MT research in the USA ceased for a long period of time.

In spite of this negative outlook, the demand for translations was growing exponentially in Canada due to the bilingual nature of the country. Research began in 1970 with the Traduction Automatique de l'Université de Montréal (TAUM) project. This project had two notable achievements: the creation of a new metalanguage and of the METEO system. The computational metalanguage was created to manipulate linguistic structures that were represented by strings and trees. In fact, this achievement was of such value that it established the foundation for Prolog, a well-known programming language that is still used to this day in NLP. In parallel, the METEO system was developed to translate weather forecasts from English to French. As such, this system was able to obtain high accuracy levels due to its "restricted range of vocabulary and grammatical structures" (Kenny, 2019, p.432).

Countries in Europe joined the MT research later on, in 1976. By then, various systems started to come into operational use, in particular the Systran MT system. This operational system was originally developed as a direct translation system between Russian and English for the US Air Force, but was later adopted by the then Commission of the European Communities (CEC). Initially, Systran was only targeted at English-to-French translations, however, as the European Communities (now European Union) started to coalesce, the need for translating other

languages increased as well. In fact, Systran was so successful that it was installed at multiple intergovernmental institutions, *e.g.* NATO, and is still thriving as of 2022.

During the 1980s, different companies from Japan started to focus their interest and efforts in MT in order to develop software for computer-aided translation (CAT). These systems were designed for microcomputers and were limited to dealing with morphological and syntactic information, "with little or no attempt to resolve lexical ambiguities" (Hutchins, 2001, p.18). For this reason, they fully depended on human assistance for preparing (pre-editing) and revising (post-editing) translations. However, Japan was not the pioneer for these microcomputers. In 1983, the Automated Language Processing Systems (ALPS) was created by developers from Brigham Young University to provide "the translator with a set of software tools to automate many of the tasks encountered in everyday translation experience" (Slocum, 1985, p.10). Hence, the ALPS system managed to resolve ambiguities in source texts and provided three different levels of assistance: multilingual word-processing, automatic dictionary and terminology consultation, and interactive translation (Hutchins, 2001). Regardless, not only did this system still required post-edition to achieve high quality translations, but it also could not account for idiomatic structures. Thus, the ALPS's recognition was rather short-lived in the MT world.

Up until this moment, automatic translations were done through rule-based machine translation (RBMT) systems. This first approach of MT required good quality dictionaries, and syntactic and semantic analysis to associate "the structure of the given input sentence with the structure of the demanded output sentence" (Okpor, 2014, p.161). The RBMT approach was known for its customizable features that allowed for full control of the translation's quality. The main reason is that each error could be corrected through a new targeted rule. Nonetheless, it still presented some limitations, specifically the fact that linguistic information still needed to be manually added and that good dictionaries are expensive and do not cover all the languages. Some examples of this approach are the PROLOG-based Logic-programming MT (LMT) system (McCord, 1985) and the previously mentioned Systran system. This second system started out as RBMT, but later transitioned to a hybrid approach that combined Rule-based MT with Statistical MT.

The Statistical MT (SMT) approach, considered state-of-the-art by the end of the 20th century, relies on statistical translation models that analyze both monolingual and bilingual corpora. As described by Okpor (2014, p.163):

> "The initial model of SMT (...) proposed by Brown *et al.* takes the view that every sentence in one language is a possible translation of any sentence in the other [language] and the most appropriate is the translation that is assigned the highest probability by the system."

SMT generates translations with the aim of identifying the relation between words, phrases and sentences in the source and target texts. Therefore, it subdivides itself into four distinguishing sub-approaches depending on its focus: word-based, phrase-based, and syntax-based. The former approach is the original model for SMT in which the translation process is decomposed in order to associate each word from a source sentence with its corresponding target. In other words, this approach consists in a word-for-word alignment task. However, these systems are quite low on overall accuracy and may not be appropriate for all languages, as they only operate at a full word level, thus excluding agglutinative languages such as Turkish (Bodrumlu, *et al.*, 2009). Phrase-based models, on the other hand, segment sentences by phrases instead of words. Note that any sequence of words can be considered a phrase if it is seen as fit, therefore this segmentation is not necessarily linguistically motivated (Koehn, *et al.*, 2003). These models begin by translating each phrase into English in order to reorder them afterwards and provide a translated output. One criticism of these two approaches is that they are not representative of syntactic aspects of languages. With this in mind, the syntax-based approach was created using parse trees as inputs, "*i.e.*, the input sentence is preprocessed by a syntactic parser" (Yamada and Knight, 2001, p.523). Syntax-based models perform three extremely useful operations on each node of the parse tree: reordering child nodes to account for languages with different word orders, *e.g.* SVO- and SOV-languages; inserting extra words to capture linguistic differences; and finally translating the overall sentence (i*bidem*).

Soon after, the fourth approach of SMT, *i.e.* hierarchical phrase-based, was created by combining the previously described phrase-based method with syntax-based models. Since these two approaches used phrases to learn how to reorder words, the hierarchical phrase-based approach aimed to use those same phrases to learn reorderings of hierarchical units that consist of both words and subphrases (Chiang, 2005).

The SMT approach, when compared to RBMT, is faster, better at detecting exceptions, and its development cost is significantly lower. However, despite these strengths, the fact that it

still requires high CPU and extra storage space, that the translation quality is unpredictable and that it does not rely on grammar rules proves that a third approach (with better translation quality, higher performance and less required investment) is essential to the progress of MT's research.

With this in mind, a new approach was developed to try and achieve higher accuracy levels and overall more control in translation by merging RBMT and SMT into a single approach, called Hybrid MT. As a result, translations can either be performed upfront through RBMT alone and then be corrected using SMT, or be pre-processed by the former and post-processed by the latter (Okpor, 2014).

The attention towards the progress in MT systems quickly increased when, in 2016, a new state-of-the-art approach and major milestone arrived - *i.e.* NMT. This new approach makes use of a single large neural network modeled to mimic the human brain to provide translation outputs for various languages and requires an abundance of training data (Tan *et al.*, 2020). NMT is capable of handling languages that are morphologically more complex and it is data-driven. In other words, it can easily meet increased demands and be customizable through multiple data sources. One problem of this approach lies in the fact that large-scale corpora are not available for the majority of language pairs (*ibidem*). Nonetheless, this approach is highly more accurate and cost efficient in comparison to the previous ones, thus justifying its current relevance in the MT field.

3.2. Translation Quality Assurance

With the worldwide rapid growth of MT, the need for evaluating translated outputs increased as well. However, asserting that a translation is good or bad is no straightforward task and, as disclosed by House (2014, p.241):

> "(...) any statement about the quality of a translation implies a conception of the nature and goals of translation, in other words it presupposes a theory of translation. And different theoretical stances must lead to different concepts of translational quality, to different ways of going about assessing (retrospectively) the quality of a translation and different ways of ensuring (prospectively) the production of a translation of specified qualities."

Evaluating translations started out as an inconsistent and somewhat groundless process, due to the fact that these assessments were done informally by bilingual professional translators and linguists. Today, however, the evaluation process is significantly more meticulous and controlled. Evaluating a translation nowadays involves checking aspects such as fluency, adequacy (between the source and the target texts), and compliance with specific requirements (*ibidem*). This evaluation can be done through two major contrasting types of quality metrics: automated metrics and manual metrics. The convenience of automated metrics to evaluate systems is of utmost importance in MT's research, due to the fact that human evaluation is often the best indicator of quality, but its cost and time-consuming tasks hinder MT's current development.

3.2.1. Manual Quality Metrics

Evaluating translation quality is still, to this day, a very intricate and challenging task, let alone in the late 1980s. At the time, translation quality was assessed by bilingual reviewers or monolingual subject experts that would provide feedback on the translated texts. However, the fact that there was no structured methodology nor any predefined tools to conduct this evaluation, led to numerous inconsistencies between translator providers. Moreover, not only were the reviewers' feedback quite vague, but the quality review steps involved in these assessments were also not well looked upon (Lommel, 2018).

To address this issue, Language Service Providers (LSPs) created translation score-cards. As described in Lommel (2018, p.111), the translation score-cards "allowed reviewers to count numbers of errors to generate overall quality scores, usually represented as a percentage, with 100% indicating no errors". Additionally, some errors were also assigned with different weights, such as *minor*, *major*, and *severe*.

Processes such as the one just described were an important step forward in the translation quality evaluation development, although they still did not account for many other inconsistencies. One of the most significant problems was related to the uncertainty of meeting customers requirements.

In the 1990s, while aiming to overcome these limitations, two new efforts for quality assessment were created. First, a new metric called SAE J2450 was developed to address automotive documentation and it featured different error types and two severity levels. However,

its intended purpose was solely for automotive service manuals and no other content type. As this metric is of specific application, it could not be applied across the board to various other texts or, in other words, it could not be generalized. The second project for quality assessment was the Localization Industry Standards Association (LISA) Quality Assessment (QA). The LISA QA Model was essentially used as the "standard for quality assessment of software and documentation localisations after its release in the 1990s" (*ibidem*, p.112). The Model featured 18 or 21 categories and, unlike SAE J2450, three different severity levels. Despite advocating transparency, both efforts failed to guarantee what they vouched for. On the one hand, the reviewers were not interchangeable and there was an inconsistency in attributing error severities, thus the inter-annotator agreement (IAA) was substantially low. On the other hand, there was an effort to standardize error types, but this decision led to multiple complications regarding the models that had been developed with specific scenarios or text types in mind. Therefore, the LISA QA Model was constantly being modified and unstable. As pointed out by Castilho (2018, p.15):

> "... if, on the one hand, this [LISA QA Model] offers a degree of standardisation, on the other it makes adaptability to the specifications of individual translation and localisation projects rather difficult."

Subsequently, two other initiatives started to actively work on assessing translation quality and developing new approaches, giving rise to the Dynamic Quality Framework (DQF) by the Translation Automation User Society (TAUS) and to the QTLaunchPad project, led by the German Research Centre for Artificial Intelligence (DFKI). As translation quality requirements change depending on a number of different factors, the DQF "provides a commonly agreed approach to select the most appropriate translation quality evaluation models and metrics depending on the specific quality requirements" (Görög, 2014, p.157). This framework provides tools to not only standardize the evaluation process, but also modify it in order to become more objective and transparent overall. In addition, a new translation error typology for use in human and machine translation analysis was created within the QTLaunchPad project. Taking into consideration various error typologies and tools such as the previously mentioned LISA QA Model and SAE J2450, this project resulted in a new and reliable error typology called

Multidimensional Quality Metrics (MQM) (Lommel *et al.*, 2014) that attempted to harmonize various evaluation methods into one master typology.

MQM is highly hierarchical as it includes different branches with dependent 'children' nodes. These branches can be distinguished depending on different issue types: *Accuracy* between the source and target texts; *Design*, *e.g.* text formatting issues; *Fluency*, *i.e.* the text's linguistic well-formedness; *Locale conventions*, *e.g.* date and time formats; adherence to *Style* guides, and finally the use of correct *Terminology*. These error types are of utmost importance for evaluating translations, because, depending on the task, some types are considered to be more important than others and the severity level of a given error may differ between *minor*, *major*, and *critical*. Thus, "*severity* refers to the nature of the error itself and its effects on usability of the translations" (Lommel, 2018, p.120).

MQM metrics were made to be "as course-grained as possible while still serving their purpose" (Lommel, 2018, p.116) when evaluating translations. To calculate an MQM score, each error is taken into account, multiplied by its severity value and its weight to generate penalty points. These points are later summed up to obtain the total and the final score, typically presented as a percentage. This way, the MQM[29] provides a better understanding of which issue types contribute to quality problems and, thereby, which issues must be addressed. Additionally, it is possible to set thresholds for different translation qualities and separate those found by the requester as acceptable from those considered inadequate.

It quickly became clear that the MQM typology and the DQF complemented each other and that it would be beneficial to merge them. So, in the follow-up project of QTLaunchPad, QT21, the integration between MQM and TAUS and DFK's efforts began. Some examples of these changes were: the initial MQM branches were restructured to match DQF's categories; DQF adopted the MQM issue names; an additional severity level - *null* - was added to allow for issues to be marked without assigning penalties to them; and two new dimensions were added, namely *Internationalization* to account for localization issues and *Verity* to "deal with the relationship of the content to the world in which it exists" (Lommel, 2018, p.118). As a consequence of this integration, the typology became smaller and overall easier to use and understand.

[29] **The MQM**: https://themqm.org/

3.2.2. Automated Quality Metrics

Automated evaluation of MT is performed by systems that provide a score regarding a given MT output. Such automated systems cannot be considered autonomous as they still rely on humans. This is due to the fact that these scores are automatically computed and based on human intervention for tasks related to collecting data, annotating or creating reference translations.

There are three types of automated evaluation according to Chatzikoumi (2020): i) metrics that provide scores in reference to a given translation; ii) quality estimation (QE) metrics; and iii) diagnostic evaluation based on checkpoints. All these metrics are used to determine if an MT system is able to meet several conditions and identify possible limitations. Therefore, an ideal automated evaluation system must be able to correlate the MT's outputs with human judgment (the most important criterion), be consistent, reliable, sensitive to nuanced errors, and handle a great range of different fields (Banerjee and Lavie, 2005).

With this in mind, Papineni *et al.* (2002) proposed a state-of-the-art evaluation metric called BiLingual Evaluation Understudy (BLEU) to measure translation performance. BLEU requires a "numerical 'translation closeness' metric" and a high quality corpus of human reference translations to estimate a translation's quality value. This value can range from 0 to 1, where 1 corresponds to a translation identical to the reference and is highly unlikely to be obtained. In other words, it compares *n-grams* of a given translation to *n-grams* of a reference and counts the number of matches. Therefore, the greater the number of matches, the better the quality of the translation and the higher the score it achieves.

Although BLEU is well known for its high correlation with human judgment, it solely relies on *precision*[30] (P), in particular on "modified unigram precision", disregarding the widely used *recall*[31] (R) metric, as its notion is unclear for BLEU's intended purposes. Hence, to tackle this and other weaknesses, Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie, 2005) was designed to explicitly match the MT output and the reference translations. In fact, it does not simply match identical words in the two strings being

[30] **Precision**: "Precision is the ratio between acceptable *n*-grams in the MT output (*i.e.* the *n*-grams also found in at least one of the reference translations) to the number of *n*-grams in the same MT output" Chatzikoumi (2020, p.5).

[31] **Recall**: "Recall is the ratio of acceptable *n*-grams in the MT output (*i.e.* the *n*-grams also found in at least one of the reference translations) to the number of *n*-grams of the reference translation (the ideal number of *n*-grams)" Chatzikoumi (2020, p.5).

compared, it can match words that are morphologically related and synonyms of each other. METEOR computes the already mentioned metrics P and R, but also "*Fmean* by combining the precision and recall via a harmonic-mean (...) that places most of the weight on recall" (*ibidem*, p.68). This way, METEOR can achieve one of the highest correlations with human judgment in comparison with other automated evaluation metrics.

The Bilingual Evaluation Understudy with Representations from Transformers (BLEURT) (Sellam *et al*., 2020) is the most recent metric for reference-based text generations focused on English. This metric uses synthetic examples, *i.e.* fit-for-purpose fabricated reference and candidate sentences, and, as a result, it is able to generalize and "model human assessment with superior accuracy" in comparison with the previous metrics described.

Around that time, a new state-of-the-art evaluation metric was developed by Zhang *et al.* (2019) called BERTScore, with the goal of evaluating semantic equivalence between candidate sentences and annotated references. BERTScore is a language generation evaluation metric based on Bidirectional Encoder Representations from Transformers (BERT), a language representation model developed by Devlin *et al.* (2019). BERTScore manages to overcome BLEU's and METEOR's limitations regarding the inability of robust paraphrase matching and capturing distant dependencies and ordering. This is done through contextualized token embeddings and matching their similarities. In other words, each token from a reference is matched to a token from a candidate sentence through contextual embeddings to compute R (and *vice-versa* to compute P) and the match between them is computed using "cosine similarity, optionally weighted with inverse document frequency scores" (*ibidem*, p.3). However, this relation with contextual embeddings may jeopardize scores, as it heavily depends on the quality of the models' embeddings.

While aiming to achieve even higher levels of correlation with human judgment, a new framework called Crosslingual Optimized Metric for Evaluation of Translation (COMET) (Rei *et al*., 2020) was developed. COMET is a state-of-the-art "neural framework for training multilingual machine translation evaluation models" (*ibidem*, p.1) and a proprietary metric of Unbabel. As such, its description has been given in *Section 2.2.1.4*. A final note, COMET has recently evolved to a compressed model, in the spirit of green artificial intelligence initiatives, called COMETinho (Rei *et al*., 2022).

3.2.3. Test Suites

With the arrival of MT, the metrics used to evaluate translation systems started to reach their limit. With this in mind, the MT community decided to look for alternative evaluation approaches from as early as the 1990s. Examples of such alternatives are the use of test corpora and of test suites to evaluate system's performances, two complementary techniques that should not be confused with each other. The main difference between them relies on the fact that the former are repositories of large amounts of possibly unrefined data, while the latter consist of a curated set of tests, representative of the structures that one wishes to analyze.

Test corpora represent "naturally occurring data, so that one can be sure that the phenomena one is testing for really do occur" (Balkan *et al.*, 1994, p.53). In other words, this technique can be used as a tool for adequacy evaluation as it depends on sequences of full sentences, *i.e.* pieces of text. Thus, it is particularly useful for testing context-dependent semantic and pragmatic phenomena. However, test corpora lack meticulousness and so the complex nature of these phenomena can make it difficult to isolate the one phenomenon being tested. Moreover, the fact that most corpora lack annotations also contributes to making the evaluation process even more complex.

Test suites, on the other hand, are lists of sentences specifically assembled to obtain a corpus of controlled examples - gold standard[32] data - used for diagnostic evaluation of a given system. As defined by Balkan (1994, p.1), test suites are:

> "... a collection of (usually) artificially constructed inputs, where each input is designed to probe a system's treatment of a specific phenomenon or set of phenomena. Inputs may be in the form of sentences, sentence fragments, or even sequences of sentences."

Given the fact that the input used for testing is checked beforehand, it is possible to control the vocabulary and the phenomenon being tested. This way the evaluator "can focus on the way the system deals with the construction without the distraction of problems relating to lexical coverage" (Balkan *et al*., 1994, p.53) and, therefore, use test suites to thoroughly evaluate systems. This evaluation method is especially useful when performing the following three major

[32] **Gold standard**: A gold standard is a data sample that accurately represents the content at issue and that has already been checked, in order to remove irrelevant and/or invalid data.

tasks, according to Balkan *et al*. (1994): i) for presenting language phenomena in an exhaustive and systematic way; ii) for generating possible combinations of phenomena; and iii) to systematically derive negative data from positive data, "by violating grammatical constraints associated with the positive data item" (*ibidem*, p.53).

3.2.3.1. Test Suites' History

Test suites have been used since the beginning of the 1990s. For instance, King and Falkedal (1990) tried to define an evaluation strategy for a bilingual translation system that would be of interest to the NLP community. As such, the authors proposed "setting up a series of systematically organized test inputs to test at least the syntactic coverage of the system". However, they quickly found a major issue related to the complex interaction between different linguistic phenomena. So, it was decided to reduce the test suites' items' linguistic complexity to an absolute minimum. This decision was found to not be as functional as predicted, as the sole task of creating test suites focused on one single language was already considered lengthy and highly demanding, not to mention regarding multiple language pairs. Therefore, two test suites were created in place of one: the first focused on a bilingual corpus representing the content the system would tackle; and the second related to translation mismatches between the source and target languages regarding lexicon and structural ambiguities. Nevertheless, it was concluded that the main virtue of test suites is their "flexibility in accounting for both the competence and the user-related performance of the system under test" (*ibidem*, p.215), only if said system includes "a substantial component of contrastively based test inputs" (*ibidem*, p.213).

The MT community has started to focus on test suites to either account for particular linguistic phenomena, as is the case with PROTEST (Guillou and Hardmeier, 2016), or to compare different MT system, *e.g.* (Burchardt *et al.*, 2017) and (Macketanz *et al.*, 2019).

The test suite PROTEST aimed to handle a specific linguistic phenomenon limitation found in the MT's evaluation process. By then, most MT evaluation methods resorted to word-alignment when generating correspondences between output words and input words. However, the problem lied in the fact that this approach would only account for content words, but disregard function words such as pronouns (in particular, anaphoric pronouns). Therefore, the test suite PROTEST was created to overcome this limitation through evaluation of pronoun correctness and ultimately improve the state-of-the-art for this linguistic obstacle.

Other work has been done regarding test suites on a more general level, namely the test suite in Burchardt *et al.* (2017), created for the language pair English-German in the QT21 project. It consisted of a balanced test set with segments from multiple corpora, grammatical resources, and lists of translation errors to compare three different types of MT systems: rule-based, phrase-based, and NMT. With the ultimate goal of automating test suites' testing, each test sentence was annotated with the phenomenon category and the phenomenon it represented. Although the results obtained were considered to be insightful on the system's performance, this test suite was a preliminary version nonetheless.

More recently, Macketanz *et al.* (2019) extended the work presented in Burchardt *et al.* (2017) by including more test sentences and better coverage of phenomena, as well as testing a significantly higher number of state-of-the-art systems.

3.2.3.2. Test Suites' Approaches

An MT system "is evaluated by how well its proposed corrections or edits match the gold-standard edits" (Ng *et al*., 2014, p.4). For this reason, test suites' construction must be done by following a specific approach that is considered appropriate for the evaluation's main goal. This is due to the fact that "if we cannot entirely trust our gold-standard data, then we cannot place too much trust in the results of evaluations carried out using that data" (Dale *et al.*, 2012, p.58).

The current section will describe different approaches in light of Balkan (1994)'s work. According to the author there are three possible approaches when constructing test suites: the bottom up approach; the top down approach; and the mixed approach.

The bottom up approach tests the system and analyzes its functions while considering them as attributes. As an example, take into consideration the case with spell checkers. Their functions consist of detecting misspelled words and providing plausible corrections, therefore their reportable attributes consist of detected misspelled words and of the corresponding correct form, found among the corrections the system proposes. Thereafter, each attribute is associated with a value, usually a percentage, which is calculated in comparison with a standard, *e.g.* test suites. However, to use test suites in evaluation of either spell checkers or MT systems in general, "the phenomena included will be of particular importance for that application" (p.3). In other words, for test suites to fulfill their purpose, they must compile representative examples of

the phenomena/content one wishes to evaluate. One example of the bottom up approach is the already described work of King and Falkedal (1990).

There are multiple system-specific test suites' construction options in the bottom up approach, as they depend on the type of required evaluation. Nonetheless, two evaluation scenarios can be distinguished: the "black box" scenario, in which the evaluator does not have access to the internal workings of the system, but is still able to test hypotheses about internal workings of the system; and, on the other hand, the "glass box" scenario, in which the evaluator has access to the system rules. In regards to the latter, the test suite's writer is able to tune his/her test suite to the rules of the system and, thus, the evaluation will often be done to test the rules' reliability. Thus, the evaluation will be of diagnostic value to locate the source of the system's error, *i.e.* a root-cause analysis. Note that even some black box system's users might still be able to have access to lexical rules and possibly intermediate representations, as is the case with the previously mentioned Systran MT system. Nonetheless, "the more access an evaluator has to a system's internal workings, the more error diagnosis he can perform" (p.4).

Unlike the bottom up approach, in the top down approach the user is not able to tune the test suite to a specialized domain. Instead, this type of test suites' construction starts with a list of linguistic phenomena, not related to any application nor system, in order to retrieve the vocabulary from a general domain. Balkan (1994) provides two examples of test suites' construction that follow the top down approach: the Hewlett Packard (HP) test suite (Flickinger *et al*., 1987); and the DITO test suite (Nerbonne *et al.*, 1992). While the HP test suite covers syntactic, semantic and discourse phenomena (focusing mostly on the former), the DITO test suite solely focuses on syntactic phenomena, but covers them in greater depth. Additionally, the author reveals one disadvantage of this type of test suite construction due to the fact that it makes it impossible to relate the test suite's phenomena with the system or application being tested.

In an attempt to combine the advantages of both approaches, Regnier and Dauphin (1994), cited in Balkan (1994), created the mixed approach, "in that the phenomena in the test suite are known to be problematic for a particular application, but are dealt with on a general level, that could potentially be used for a range of applications" (*ibidem*, p.5). This is only possible as the mixed test suite construction approach starts by identifying test sentences that are problematic for translations, then uses them to create generic test sentences, in order to isolate

phenomena and, thereafter, associate phenomenon with other illustrative sentences. Thus, for all variations of one phenomenon, an exhaustive list of test sentences are written.

3.3. MT's improvement with Grammatical Error Detection and Correction

Despite all the investment in MT's research and development so far, systems cannot yet guarantee sentence's fluency and coherence in MT outputs. Although there is also a general consensus regarding the automation of grammatical error detection (GED) and correction (GEC), most work on error detection and correction is focused on specific types of errors such as prepositions and determiners. Thus, there has been limited work on systems that could handle all error types at once.

3.3.1. GED

For an MT system to accurately correct translation errors, it needs to properly identify those errors beforehand. Hence, GED systems become involved in order to take potentially erroneous sentences as input and identify the tokens that do not follow certain linguistic rules, *i.e.* that are grammatically incorrect. However, this detection task is a quite complex process as it involves various NLP tools and dependencies between tokens. Take the following example into consideration:

(1) *This are a gramatical sentence.

We can pinpoint the errors in this sentence solely through intuition, without explicitly knowing grammatical and language rules. On the other hand, that is not possible for a system as it is not capable of intuitively making decisions, but instead relies on a multitude of explicit universal dependencies, word classifications and language rules to detect errors.

The first step in error detection is tokenization, or in other words, splitting the sentence into words/tokens. This is a relatively simple process for languages such as English, since words are separated from each other with whitespaces. However, for languages such as Chinese, the tokenization task is a lot more complex, as there is no clear separator between words.

Another crucial step in error detection relies on parsing the sentence, *i.e.* breaking a sentence into its component parts. Each token can be classified according to the POS it belongs to. Therefore, the tokens from the example given above may be classified as:

(2) DET VERB DET ADJ NOUN

It is also possible to do an even more in depth analysis for each token by detailing that the first determiner is a demonstrative, that the verb is conjugated using the 3rd person singular in the present, and so on. Through this analysis, the system can check if words agree in number, gender and person with one another. At this point, the system would be able to identify in example (1) one spelling error regarding *'*gramatical*' and an inconsistency between the determiner and the verb, thus classifying it as an agreement error. Thus, parsing tools can be used to analyze large volumes of text without human interference, to reveal error patterns and other issues that may be overlooked otherwise.

While some error detection systems rely on corpora as reference, others do not. Tezcan (2017) states that "the detection of grammatical errors in MT output, without relying on reference translations, can be considered as a QE task that caters for a particular MT error type" (*ibidem*, p.134). This is due to the fact that by calculating the number of errors detected in a text in relation to the total number of words, it is possible to estimate the post-editing effort at segment level to filter low quality translations (Specia *et al.*, 2009) and identify which error types are the most problematic.

Rei and Yannakoudakis (2016) proposed a framework for token-level error detection using neural compositional models. Following the long-short term memory (LSTM) neural network by Hochreiter and Schmidhuber (1997), the authors created a bi-directional framework, Bi-LSTM, in which each sentence's token was labeled as "correct" or "incorrect". This binary sequence labeling achieved state-of-the-art results by capturing dependencies over longer distances. Soon after, Kaneko *et al.* (2017) extended Rei and Yannakoudakis' research by adding grammaticality and error patterns to their work, thus improving GED accuracy.

Over time, there has been an effort to identify all error types at once in texts using GED. However, this was not always the case. In 2011, a pilot shared task was carried out by Dale and Kilgarriff (2011) entitled Helping Our Own (HOO) that focused on manually detecting, identifying and correcting errors from non-native English speakers. By analyzing fragments of texts from publications of the Association for Computational Linguistics, different teams focused on analyzing a series of error types. However, due to the task's manual nature, the cost of data annotation was considerably high and very labor intensive. To overcome this, Dale *et al*. (2012) carried out the HOO 2012 shared task, focused on preposition and determiner error detection and

correction. Instead of trying to cover such a wide range of error types, it was decided that it would be beneficial to target solely a reduced number of them.

The following year, the Computational Natural Language Learning (CoNLL) shared task (Ng *et al.*, 2013) continued to work with data from students of English as a second language focusing on five error type categories: determiners, prepositions, noun numbers, verb forms, and subject-verb agreement. The main goal was to detect and correct errors regarding natural language processing, named entity recognition, semantic role labeling, dependency parsing and coreference resolution. Their work was extended the year after with the CoNLL 2014 shared task (Ng *et al*., 2014), in which the focal point was changed to the evaluation of algorithms and systems for automatically detecting and correcting errors of all types.

In an effort to combine both HOO's and CoNLL's work, the Building Educational Applications (BEA) shared task (Bryant *et al.*, 2019) not only provided a platform where systems could be evaluated under more controlled conditions, but it also introduced a new annotated dataset with more diverse examples, that allowed systems to better generalize unseen data and achieve better results.

Kasewa *et al.* (2018), focusing on error detection research, decided to supplement existing manual annotations with synthetic instances in an attempt to improve GED accuracy. In other words, using NMT to learn the distribution of errors in texts and to generate appropriate errors of various types, the authors were able to improve overall systems' metrics and achieve even better results.

Thus far, the majority of error detection research focused on English. Therefore, Tezcan *et al*. (2016) decided to detect errors using dependency parsing and treebank querying for MT outputs in Dutch. Representing the abstract grammatical relation between constituents with these structures, the authors were able to overcome the MT's limitation of requiring manual correction when producing grammatically correct sentences. A few years later, Zhao *et al*. (2018) decided to focus on detecting grammatical errors in essays from non-native Mandarin Chinese speakers instead, by carrying out the Natural Language Processing and Chinese Computing (NLPCC) shared task. As most involved teams treated error correction as an MT task, it was made possible to automatically return the corrected texts and broaden GED and GEC's research field.

GED can be considered a low/mid-resource task, as the total number of correct tokens in a text often outweighs the number of incorrect ones (Leacock *et al.*, 2014). However, as stated in

Bell *et al.* (2020), GED is a monolingual (statistical or neural) MT problem nonetheless. In order to extend the current state of the art for error detection, Bell *et al.* (2020) decided to incorporate contextual embeddings to the Bi-LSTM framework already mentioned above. The authors demonstrated that by using word representations previously constructed based on the context the words appear - *i.e.* contextual embeddings -, it was possible to substantially improve GED performance and, consequently, achieve new state-of-the-art results. Thereby concluding that "contextual embeddings bring the possibility of leveraging information learned in an unsupervised manner from high volumes of unlabeled data, by large and complex models, trained for substantial periods of time" (*ibidem*, p.7).

Recently, Yuan *et al*. (2021) achieved significant improvements in binary GED and new state-of-the-art results on different benchmark datasets. While demonstrating the influence of multi-class GED, the authors also proposed a new multi-encoder GEC model for the English language. In order to achieve this, they used pretrained transformer-based language models - *e.g.* ELECTRA (Clark *et al.*, 2020) -, and extended them to multi-class detection using different error type tagsets. Hence, overcoming the previous system's limitation of attaining high accuracy results solely when detecting specific error types.

3.3.2. GEC

Once GED systems fulfill their assigned task of identifying errors within the input text, the following step is the sole responsibility of GEC systems. They must accurately correct the identified errors, while maintaining the original meaning of the sentence intact.

To train GEC models, two types of datasets can be used as described in Ailani *et al*. (2019): error-coded text or parallel datasets. The former consists of texts with errors generally coded by human annotators, while the latter combines the raw original text with its corresponding reference, a corrected version of the text with no explicit errors identified.

GEC's research has achieved growing relevance in the MT field. Currently, there are three major approaches when studying these systems: rule based approach, classifier based approach, and MT approach. The latter can be further subdivided into SMT and NMT. The rule based approach consists in hand-coded grammar rules with simple pattern matching (Bustamante and León, 1996), and it is mainly focused on syntactic analysis. Although it is considered to be relatively simple to implement, it cannot detect complex errors nor generalize well. This is due to

the fact that it is not possible to create rules that account for every combination of errors in a text (Ailani *et al.*, 2019). The second approach mentioned, *i.e.* the classifier based approach, considers words as class labels and *n-grams*, POS tags, and grammatical relations as features. This approach has two main limitations: it assumes each error as being independent (which is not always the case, *e.g.* context dependent issues) and, on the other hand, each classifier is only capable of detecting one single type of error while considering the remainder of the sentence to be error-free. To overcome such limitations, Rozovskaya *et al*. (2013) built multiple classifiers, each responsible for correcting one error type, all combined into one single pipeline. Nonetheless, the classifier based approach's limitation of not performing well with dependent errors still remains. The third and last approach on GEC's research concerns different MT models and systems. On the one hand, SMT models are used to solve all error types, instead of focusing on a single one. Each error is classified within two types: either it is an error that can be solved efficiently solely by increasing corpus size, or it is an error dependent on contextual information (Mizumoto *et al.*, 2012). On the other hand, NMT systems are developed using an encoder-decoder mechanism (Chollampatt and Ng, 2018), where the encoder reads the sentence and encodes it into a vector, and the decoder outputs a translation, based on the encoders vector and on previous predicted words.

In an attempt to further improve GEC's research and achieve state-of-the-art results, Grundkiewicz and Junczys-Dowmunt (2018) created a hybrid approach between SMT and NMT. According to the authors, the MT's output would be corrected by SMT and passed as input to NMT. In this manner, it would be possible to increase recall and, consequently, improve GEC's overall performance.

4. Methodology

The main purpose of the automatic evaluation metrics is to compare the output of an MT system with human-reviewed translations, by measuring how close this MT output is to the reference translation. One way to achieve this is through the use of test suites or, in other words, by relying on good quality annotations. Smartcheck depends heavily on annotations and on their typology for classifying errors in order to provide suggestions that mirror those of editors. The current section aims to find Smartcheck's strengths and weaknesses in light of a data curation process developed along the work conducted with this dissertation, in order to improve overall machine translation quality.

The translation quality assessment presented in the current section merges the reliability of verified gold standard annotations with the time and cost effectiveness of an automated system. Thus, this methodology consists in the evaluation and analysis of Smartcheck's grammar-checking rules in production, through the creation of test suites built on gold standard annotations to cover every language pair Unbabel supports.

The current study seeks to respond to the following research question, which emerged upon analyzing the results from the first baseline evaluation: Is the dataframe being used to evaluate Smartcheck rules, *i.e.* the test suites, appropriate for its purpose?

In an attempt to answer this issue, the methodology was divided into three main stages:

1. Baseline analysis of previous Surfboard rules
2. Root-cause analysis regarding the results obtained from the previous analysis
3. Creation of a new and improved evaluation dataframe following the bottom down approach, with curated data and revised annotations, *i.e.* creation of new test suites following a reliable and methodological process.

We have chosen to refer to the previous and new test suites as 'previous evaluation dataframe' and 'new evaluation dataframe', respectively. Moreover, the data analyzed in this study was produced by different MT systems proprietary to Unbabel. Unbabel's MT engines are transformer-based models (Vaswani *et al*., 2017) trained with the Marian toolkit (Junczys-Dowmunt *et al*., 2018).

The following will be discussed in this chapter: *Section 4.1* will describe the pilot analysis conducted to assess the Surfboard rule's precision and effectiveness when detecting translations issues. The results obtained upon evaluating each rule will then be presented in

Section 4.1.1. Due to the fact that the results were not up to expectations, a root-cause analysis was conducted in *Section 4.2* to identify the possible underlying causes of the problem. Hereby, we hypothesized that the low metric values could be the result of an inadequate evaluation standard, *i.e.* a faulty evaluation dataframe. Therefore, *Section 4.2.1* and *Section 4.2.2* describe the methodology used to assess the dataframe's suitability. The former consisted in the preparation of the data for ease of convenience regarding the subsequent section, where the dataframe analysis was in fact performed. Once the annotations revision was completed, it was concluded that creating a new evaluation dataframe with up-to-date and revised examples would be the best approach to achieve higher metric values. Thus, *Section 4.3* is dedicated to the creation of a new evaluation dataframe for the assessment of Surfboard rules. First, the data retrieved consisted of current content translated by Unbabel's MT systems and so it was important to filter out irrelevant data and organize the remaining elements (the data curation process is described in *Section 4.3.1*). The annotation curation methodology followed is thoroughly explained in *Section 4.3.2*, where every criteria applied is defined along with various examples. Finally, the analysis' results are shown in *Section 4.3.3*.

4.1. Previous Surfboard Rules' Evaluation

Firstly, it is important to notice that a baseline analysis was conducted on Smartcheck and its associated rules, reason for dedicating a subsection of the current chapter to the obtained results and for not including them in the following chapter. We have chosen to present the baseline results in our methodology since the root-cause analysis of the outcomes guided our steps in designing and driving our methodological procedures. The pilot analysis or baseline analysis is of crucial importance for the decisions guiding how to structure the new evaluation dataframe (EDF).

The core of the study is based upon the results derived from the evaluation of the Surfboard built-in quality metrics. In this regard, and to better understand what needs to be improved in the system, thirty nine enabled rules in the platform - by November 5th, 2021 - were both validated and evaluated.

Surfboard rules were created to account for translation issues regarding a variety of different target languages. *Figure 10* compiles the number of language rules on Surfboard that were evaluated for the current baseline analysis. For some rules to be applied, it was required

that a translation followed a particular register - formal or informal -, while other rules were created to be applied to translations in general. In other words, for each accounted target language, there could be either one or more of: a general language rule; a rule to be applied specifically in formal registers; and another rule for informal registers. Note that for every language pair, the source language was always English (EN).

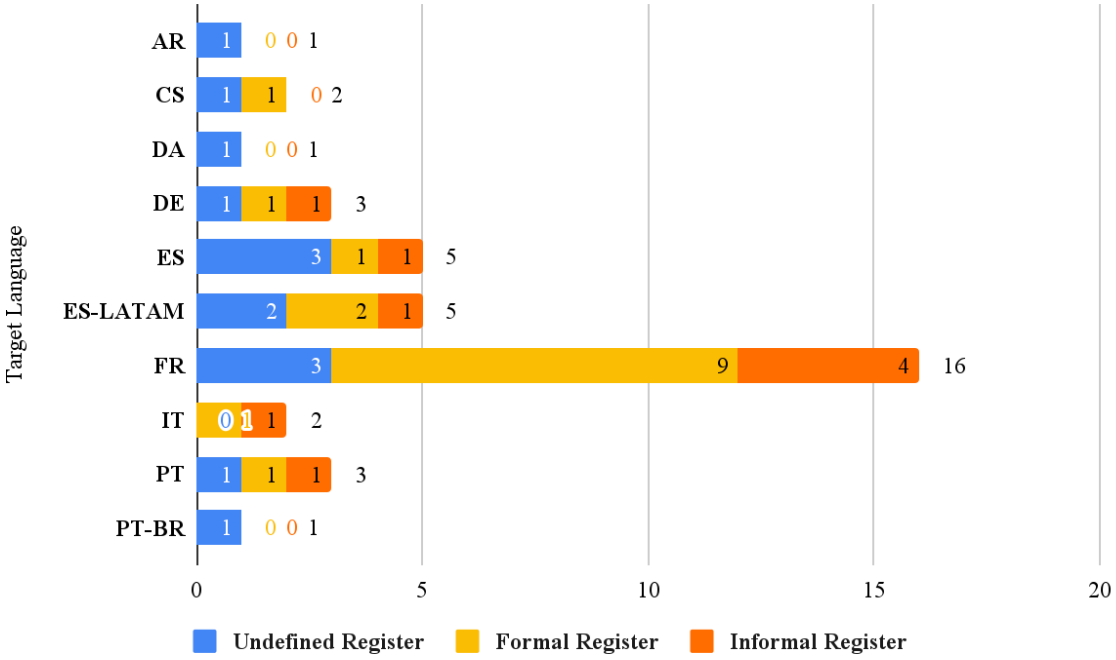


Figure 10. Surfboard Rules: Number of rules per Target Language

The following examples show the difference between each type of rule mentioned above:

(1) Undefined Register

Target language: DA; Category: Punctuation

Rule: No punctuation mark must be used after the greeting.

(2) Formal Register

Target language: IT; Category: Lexical Register - Formal

Rule: Must use "Cordiali saluti" or "Distinti saluti" in closings

(3) Informal Register

Target language: IT; Category: Lexical Register - Informal

Rule: Must use "Ciao", "Saluti" or "Arrivederci" in closings

4.1.1. Surfboard Rule's Evaluation Results

Every rule in production was possible to evaluate, leading us to the conclusion that the syntax used to create them was correct. However, the overwhelming majority of these rules revealed severely low metrics, shown in *Table 1*. Nonetheless, it is worth mentioning that the obtained values in the current section are specific to each rule's individual performance and not regarding a set of translated sentences. In other words, the metric values will increase as soon as said rules were to be applied to erroneous tokens in a text. For instance, consider a rule that covers spelling issues regarding a specific word to have 0.5 for Precision. If said rule is triggered in a text multiple times, its P value will increase the more the token is correctly detected. Thus, the values obtained in evaluations such as the current one will inevitably be lower than those obtained in a set of translations.

In fact, as it is summarized in *Figure 11*, seventeen different rules scored 0 for every metric, *i.e.* for P, R and F1-score, and nine other rules resulted in a production time out, so no values could be obtained. The remaining thirteen rules that were evaluated achieved extremely low values for each metric, with no rule scoring higher than 0.1 for F1-score and R except for rule *11*, and solely three rules scoring higher than 0.4 for P.

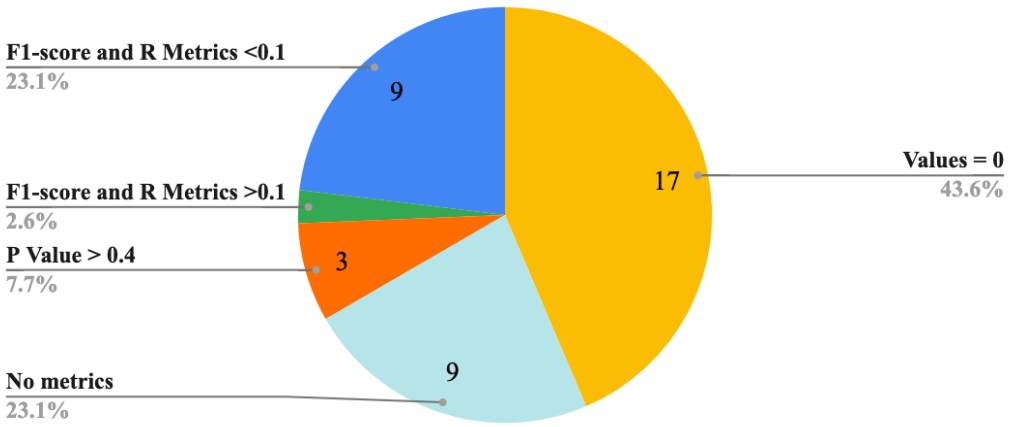


Figure 11. Surfboard Rules: Baseline Results Summary

Target Language	Rule ID	P	R	F1
AR	**117**	No metrics		
CS	**118**	No metrics		
	125	No metrics		
DA	**115**	No metrics		
DE	**2**	0.477	0.046	0.083
	3	0.089	0.078	0.083
	12	0	0	0
ES	**4**	No metrics		
	5	No metrics		
	35	No metrics		
	73	No metrics		
	91	No metrics		
ES-LATAM	**28**	0	0	0
	32	0.078	0.013	0.022
	33	0.133	0.029	0.047
	110	0.25	0.002	0.005
	113	0.014	0.001	0.001
FR	**7**	0.04	0.001	0.002
	44	0	0	0
	45	0.123	0.073	0.092
	46-58	0	0	0
IT	**8**	0.026	0.005	0.008
	9	0.156	0.063	0.09
PT-BR	**25**	0.474	0.013	0.026
PT	**10**	0.002	0.001	0.001
	11	0.448	0.103	0.168
	31	0	0	0

Table 1. Surfboard Rules: Baseline Analysis Results

On the other hand, the evaluation's outcome also provided values for how many times the rules were triggered (correctly - cases of TPs - and incorrectly - cases of FPs) and how many times errors were annotated, but no rule was able to account for them and thus did not fire - cases of FNs. For an MT system to achieve high metric values, cases of FPs and FNs are expected to be reduced when compared to those of TPs. However this was not the case as there were significantly less cases of TPs than any other. This is shown in *Figure 12*, where the *y* axis accounts for each time a specific rule (each associated to a correspondent ID in the *x* axis) is fired or should have been fired, but it was not.

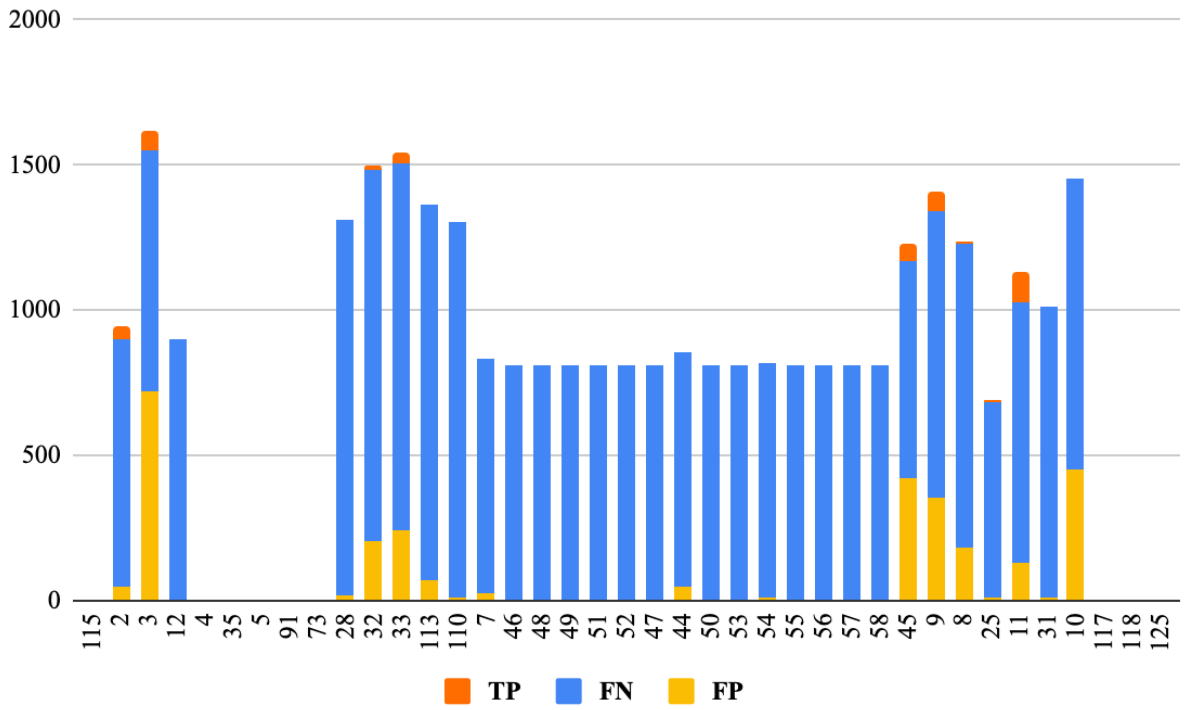


Figure 12. Surfboard Rules: TPs, FNs and FPs per rule

Therefore, the results obtained from the baseline analysis show severely low metric values. To understand the reason behind Surfboard rules not flagging existing errors in MT outputs or incorrectly flagging correct tokens, a pilot analysis was performed to identify the main possible issue.

4.2. Root-cause Analysis for the Low Evaluation Metrics

The EDF contains annotated data for multiple language pairs on both registers (formal and informal) and it must be considered the gold standard of annotated *tickets*, with fit for purpose and representative examples of the content the systems translate in order to properly evaluate error detection systems. This is due to the fact that, as previously mentioned, Smartcheck relies on annotations to give out suggestions to editors regarding possible errors in *ticket* translations. However, as shown in the *Section 4.1.1*, Surfboard's rules failed to meet quality expectations.

The first research question was raised after looking into the results of the baseline analysis and aiming to find where the main problems lie. For this reason, the methodology follows a root-cause analysis type of approach, in which the influencing factors are identified and further analyzed. The hypothesis is that the results gathered may be affected by multiple factors, and particularly by the evaluation standard currently being used.

For any evaluation set to fulfill its purpose, the data must be periodically updated, so that we can assure its suitability. It is expected for a business to adjust to different situations and conditions as it expands on the market. Thus, variables such as new clients, products, and provided services are likely to change over time, culminating in a constant need for an evaluation of quality, for an improvement on accuracy and revision of mistakes. With this in mind, the annotations in the EDF must be revised to constantly represent the annotations' gold standard. Therefore, it can be assumed that one reason for the low metric values is that the data included in the EDF does not fully represent current Unbabel's translations and, more specifically, it is not representative of the *tickets* translated nowadays. Hence, the EDF used can be considered as partially outdated.

Another reason to justify the need for updating the EDF's data is related to the fact that the common errors found in translations are constantly changing. When certain expressions and structures are problematic for MT systems, they must be recognized in order for a solution to be found. However, identifying errors and solving them is a continuous process, because new issues are constantly arising and retraining the MT systems frequently involves creating new data.

All things concerned, we hypothesize that the low metrics obtained may be substantially related to the evaluation metric being used for Surfboard rules' evaluation. Hence, the first research question is related to the EDF's suitability.

With that in mind, the dataframe used was analyzed by considering six aspects: i) the translation step (if the translation was provided only by an MT system or if the MT output has been edited and the text is the final translation[33]); ii) segment's duplications; iii) correct annotations (there was an issue with the MT output and it was correctly identified and classified), iv) incorrect annotations (there was no MT output related issues, but the word/token was considered as having an error) and v) missing annotations according to the standard asserted by typologies and guidelines; and, finally vi) the severities (*minor*, *major* or *critical*) associated with each translation error. As such, the analysis was divided into two different steps:

1. Preparing the EDF for annotation revision by sorting it according to language pair and register
2. Annotation Revision according to the previous Unbabel Annotation Error Typology (version 2) and Language Guidelines.

Note that the first step in the EDF's analysis served the sole purpose of assisting the annotation's revision process, so no annotations were corrected thus far.

4.2.1. EDF's Preparation

Prior to the actual analysis, the dataframe was filtered and prepared to have its focus solely on the relevant data for the research, hence the need for a curation process. The EDF consisted of *tsv* (tab separated values) files with information regarding the translation step, source and target languages regarding associated segments, source and target positions and tokens, as well as target error tags and severities. The first step on the EDF's analysis was to concatenate all files into one single dataframe using VSCode[34] along with pandas[35], as shown in *Figure 13*. For each file, an alias was created for functionality purposes with a corresponding number. Thus, one file was named *df1*, another one was named *df2* and so on until *df7*. This action was done using the following:

[33] **Final Translation**: When the target text has already been translated by MT and, if needed, reviewed by the editor's community. The final translation is the translated text that the customer will receive.

[34] **VSCode**: A source code editor with features to support debugging, syntax highlighting, intelligent code completion, snippets, and code refactoring. For more information, please refer to: https://code.visualstudio.com/

[35] **Pandas**: An open source Python package widely used for data analysis and machine learning tasks, such as data cleaning, data fill, merges and joins, statistical analysis, data visualization, and more. For more information, please refer to: https://pandas.pydata.org/

df*number* = pd.read_csv("*file_name*", delimiter='\t')

The next step in the data curation process was to merge all files into one single dataframe, titled *df_out*. This was done by using the expression:

df_out = pd.concat([df1, df2, df3, df4, df5, df6, df7], axis=1)

```
In [4]: df1 = pd.read_csv("meta_data.tsv", delimiter="\t")
In [5]: df2 = pd.read_csv("source.positions.tsv", delimiter="\t")
In [6]: df3 = pd.read_csv("source.tokens.tsv", delimiter="\t")
In [7]: df4 = pd.read_csv("target.error_tags.tsv", delimiter="\t")
In [8]: df5 = pd.read_csv("target.positions.tsv", delimiter="\t")
In [9]: df6 = pd.read_csv("target.severity_tags.tsv", delimiter="\t")
In [10]: df7 = pd.read_csv("target.tokens.tsv", delimiter="\t")
In [11]: df_out = pd.concat([df1, df2, df3, df4, df5, df6, df7], axis=1)
```

Figure 13. Data curation process: Merge of files into one dataframe

The subsequent data analysis was done resorting to ModernCSV[36], and so the file type was changed from *tsv* to *csv* (comma separated values), with: **df_out.to_csv("df_out.csv")**. From here on, *df_out* was renamed as *df* and it was filtered in order to separate final translations from MT outputs, as illustrated in *Figure 14*.

```
In [15]: mt = df[df.translation_step == "mt"]
In [16]: mt.to_csv("mt.csv")
```

Figure 14. Data curation process: Filtering of translation step

[36] **ModernCSV**: A tabular file editor/viewer application used to analyze data, check files for uploading to databases, modify configuration files, maintain customer lists, and more. For more information, please refer to: https://www.moderncsv.com/

Separating translations steps was crucial in the curation process, due to the focus of this research being on Smartcheck, a tool applied only to MT outputs. From then on, *mt* was filtered to separate registers and two new files were created, as shown in *Figure 15*, namely *mtF* and *mtINF* for the formal and informal registers, respectively.

```
df = pd.read_csv("mt.csv")
mtF = df[df.tone == "Formal"]
mtF.to_csv("mtF.csv")
mtINF = df[df.tone == "Informal"]
mtINF.to_csv("mtINF.csv")
```

Figure 15. Data curation process: Filtering of registers

This way, it was possible to look into the target texts and respective annotations and severities more conveniently. Thereby, the EDF's analysis began by looking into each language's content and adding new auxiliary columns in Google Sheets to assist in the data verification process. All modifications to the EDF mentioned so far served the purpose of averting possible mistakes, given that the review of the annotations was to be done manually, following the Unbabel Annotation Error Typology. These auxiliary columns consisted of four columns regarding each annotation per segment/token and two new ones regarding the associated severities. The former were as follows: one to account for the total number of current annotations; two other columns for the total number of correct and incorrect ones; and the last one to account for missing annotations. As for the columns concerning severities, the additions made followed a similar arrangement: one new column to account for the total number of existing severities; and another new one for the total number of correctly attributed severities.

These changes in the EDF allowed for a more thorough analysis, both for the errors' classification and for the severities. This is especially true in regards to the latter, because the number of total severities must be equal to the total number of correct annotations.

4.2.2. Annotation Revision

Once the step of preparing the EDF for revision had been completed, the annotations had to be reviewed. Only seven different languages were revised due to time constraints - namely,

German (DE), Spanish (ES), Latin American Spanish (ES-LATAM), French (FR), Italian (IT), European Portuguese (PT), and Brazilian Portuguese (PT-BR) - and the total number of annotations per register was fairly equal to each other, with a total of 4358 annotations for the formal register and 4297 for the informal one. *Tables 2* and *3* show the distribution of correct, incorrect and missing annotations per target language, as well as information regarding associated severities and the total number of duplicates in the dataframe. Moreover, these duplications were collected using the following expression:

=IF(COUNTIFS(*source_text_column$cell:cell*, *cell*, *target_text_column$ cell:cell*,*cell*)>1, "DUPLICATE", "-")

	Target Language	Total Ann.	Correct Ann.	Incorrect Ann.	Missing Ann.	Correct Severities	Incorrect Severities	Duplicates
Tone: Formal	**DE**	555	498	57	49	448	50	157
	ES	15	13	2	2	12	1	0
	ES-LATAM	1104	609	495	122	589	20	84
	FR	575	511	64	125	332	179	128
	IT	893	650	243	154	229	421	207
	PT	541	453	88	53	72	381	36
	PT-BR	675	519	156	212	488	31	277
	TOTAL	**4358**	**3253**	**1105**	**717**	**2170**	**1083**	**889**

Table 2. EDF's Analysis: Totals - Tone: Formal (Note: "Ann." stands for "annotations")

Due to the abundance of duplications and of incorrect annotation and severities, it can be concluded that there is a considerable amount of inaccurate and noisy data in the EDF. Additionally, even some target text segments contained errors and, as such, the issues in their respective target were not related to the annotations done, but to the source itself. Removing noisy data from evaluation sets is of crucial importance, as "when noise is present, the classifier built from this data would be less accurate a description of the target concept and thus be of a lower utility" (Teng, 1999). Ergo, the research question raised previously regarding the

suitability of the EDF used to evaluate Surfboard rules can now be answered. Not only are there copious amounts of data that needed improvement, but also the existing repetitions are detrimental for data-based systems, since a "larger amount of memory will be required to store the data, as well as the complexity of the data, will increase" (Bhoi *et al.*, 2017).

	Target Language	Total Ann.	Correct Ann.	Incorrect Ann.	Missing Ann.	Correct Severities	Incorrect Severities	Duplicates
Tone: Informal	**DE**	1424	1370	54	139	1309	61	155
	ES	75	67	8	3	65	2	0
	ES-LATAM	798	572	226	127	534	38	105
	FR	202	198	4	227	160	38	4
	IT	781	525	256	90	400	125	251
	PT	762	730	32	81	481	249	161
	PT-BR	255	211	44	114	191	20	257
	TOTAL	**4297**	**3673**	**624**	**781**	**3140**	**533**	**933**

Table 3. EDF's Analysis: Totals - Tone: Informal (Note: "Ann." stands for "annotations")

Instead of reviewing annotations and severities belonging to a dataframe loaded with duplicated and unrevised data, it was decided that it would be better to do so on more recent annotations. In the interest of being time-efficient and due to the fact that every segment had to be reviewed regardless, the existing EDF was dismissed.

In sum, the noisy and deprecated data definitely needed to be upgraded into a more defined and revised dataframe. As such this process will be thoroughly explained in the following section.

4.3. Test Suites' Construction - New Evaluation Dataframe

The following step in the methodology was to create a new and improved EDF following the bottom down approach referred to in Section *3.2.3.2*, due to the inadequacy and problems found in the previous one. In order to accomplish this, an entire batch of annotations for every

language pair Unbabel supports was retrieved from Jupyter Notebook by filtering out solely the relevant data - MT outputs. The new EDF's creation consisted in two consecutive operations:

1. Compiling already annotated segments into one single batch
2. Reviewing annotations to remove incorrect categorization of MT errors and annotate overlooked issues.

It is worth noting that the new EDF does not contain gold translations, *i.e.* translations in which the existing error has been corrected, since it would be out of scope of the work presented here.

Although the new EDF follows a similar approach as described in Avramidis *et at*. (2019), our work is conducted by manually identifying and classifying errors through annotations. As such, no error type was identified using regular expressions and no data was augmented - as all EDF's content comes from real data, retrieved from messages sent by in-house agents.

4.3.1. Data Curation

Similarly to the process of curating the data from the previous EDF, the data retrieved from Jupyter Notebook was collected from January 1st, 2021 to February 28th, 2022 and it was filtered using pandas (refer back to *Section 4.2.1*) to separate registers - *Figure 16* - and sorting out target languages - *Figure 17*. Afterwards, new files were created for each language pair (lp) using the following expressions:

df = *register*
registerTARGET_LANGUAGE* = df[df.lp == "*language_pair"]
***registerTARGET_LANGUAGE*.to.csv("*registerTARGET_LANGUAGE*.csv")**

This separation was done purely for operational purposes, due to the fact that manually reviewing annotations requires a higher level of effort and control over the data.

```
In [8]: formal = df[df.register == "formal"]
In [9]: formal.to_csv("formal.csv")
In [10]: informal = df[df.register == "informal"]
In [11]: informal.to_csv("informal.csv")
```

```
In [19]: df = formal
In [20]: formal = df.sort_values("lp")
In [21]: formal.to_csv("formal.csv")
In [22]: df = informal
In [23]: informal = df.sort_values("lp")
In [24]: informal.to_csv("informal.csv")
```

Figures 16 and 17. New EDF: Filtering data by register (left Figure) and sorting language pairs (right Figure)

The process of reviewing annotations was constant throughout all languages. New columns were added to facilitate the verification of total severities, similarly to the modifications done when analyzing the previous EDF. In addition, all duplicates were identified from the dataframe using the same expression referred previously, as well as filtering values with "DUPLICATE" and removing them.

Upon cleaning the EDF, some annotations were found to be context-dependent, *i.e.* they needed information beyond their segment, and that limitation was not being taken into account. To overcome this, annotations that involved document-level information were analyzed and it was concluded that the following error types were either fully or occasionally conditioned by context: *Inconsistency*, *Capitalization* and *Agreement*. As for *Inconsistency* annotations, all errors classified as such were found to always be susceptible to nearby segments, while the remaining two only relied upon context in particular instances. Regarding *Capitalization*, this dependency only existed in greetings and closing due to language specificities for *tickets*. For instance, in German and Finnish, if a comma is used in the greeting, the following sentence starts in lowercase, but if an exclamation mark is used, the following word must be capitalized. The third and last error type that depends on context is *Agreement*. However, the majority of these annotations do not depend on context and thus it is important to define which ones actually have such limitations. Only segments containing one *Agreement* annotation depend on context. For example, the token "*encantada*" (adj. fem. sing.) in "*Estaré más que encantada de ayudarle*." was annotated as *Agreement*, while referring to a male speaker, mentioned later in a forthcoming segment.

After this analysis was completed, auxiliary columns were added to the EDF to account for each one of the occurrences mentioned above. Therefore, to find instances of *Inconsistency*, the following expression was used:

=REGEXMATCH(*typology_error_column*,"inconsistency")

Identifying context-dependent *Capitalization* errors involved three separate operations, in that the first one (A) was meant to retrieve the first tokens from two different columns - target text and error -, the second one (B) verified if the first annotated error corresponded to a capitalization error, and finally a third column to check if both of these conditions were true (C). In other words, the first word of the translated segment must be that segment's first error as well as be annotated as *Capitalization*. Hence, the need for the following expressions:

(A) =REGEXEXTRACT(*target_text_column*,"[\w]*")=REGEXEXTRACT(error_column,"[\w]*")
(B) =REGEXEXTRACT(*typology_error_column*,"[\w]*")="capitalization"
(C) =IF(*result_from_(A)*=TRUE,*result_from_(B)*=TRUE)

In relation to *Agreement*, due to the fact that a segment must only contain one error of this type to be considered as context-dependent, the following expression was applied:

=IF(COUNTIF(*typology_error_column*,"agreement")=1, "yes", "no")

Lastly, a final column entitled "needs_context" was added to verify if one or more of these conditions occur. For that last verification, the following was used:

=IF((OR(*inconsistency_result*=TRUE, *capitalization_result*=TRUE, *agreement_result*="yes")), "yes", "no")

Finally, the last operation in the creation of a new EDF was the combination of all files' data. Auxiliary verification columns were excluded, formal and informal registers were merged together, every language pair was combined into one single dataframe, and finally the values from the "needs_context" column were added. This last step is of most importance because, although Smartcheck can check for context-dependent errors at a document level, SURF rules

cannot do so thus far. This last step concludes the creation of the new EDF and opens the opportunity to meticulously curate data and review every annotation.

Thus, the new and curated EDF shall include balanced formal and informal segments, where some may be considered context-dependent while others are not. The dataframe, as shown in *Figure 18*, consists of a total of 27 511 segments in which less than half were annotated and only 2,5% (a total of 693 segments) are context-dependent, according to the totals retrieved from the expressions above.

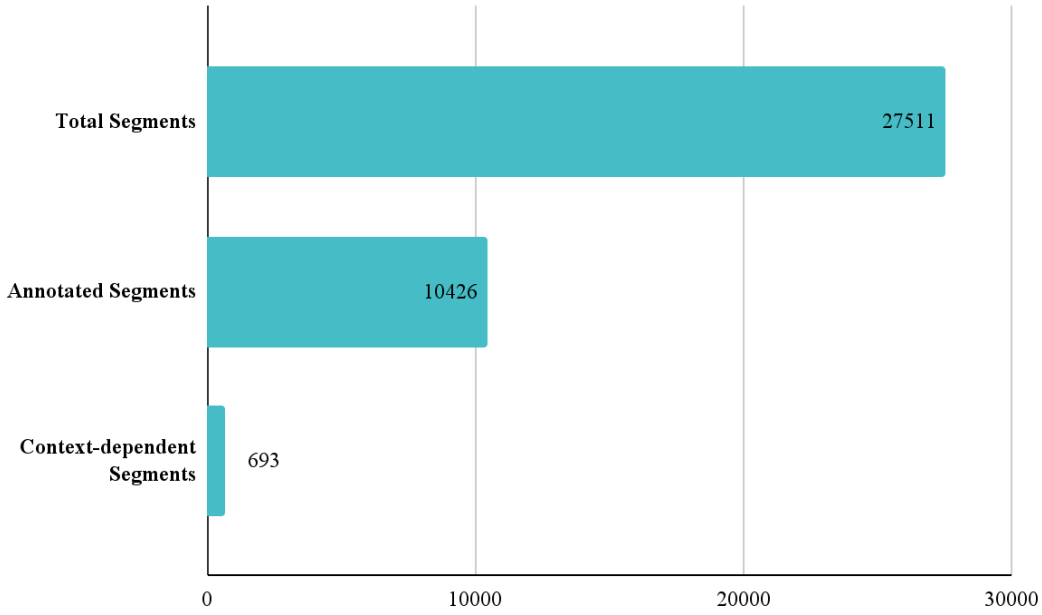


Figure 18. EDF's Content: number of total segments, annotated segments and context-dependent segments

4.3.2. Annotation Curation

Once the preparation of the dataframe was complete, the following step in the data curation process consisted in the analysis of annotations and associated severities. This analysis was based on Annotation and Language Guidelines created by Unbabel, prioritizing the revision of annotations and severities related to *Orthography*, *Punctuation*, *Register* and localization challenges, *i.e.* locale conventions, such as date and time formats, currency and numeral related issues. Note that for some target languages, the revision was done in more detail by three different linguists and translators, in particular for DE, ES, ES-LATAM, FR, IT, JA, KO, PT,

PT-BR, ZH-CN, and ZH-TW. This is due to the fact that even "assuming that we agree an error is present—and this is not always in itself straightforward—there is often more than one way to correct that error" (Dale *et al*., 2012, p.58). Cases of ambiguity related to error categorization were also discussed throughout the revision process to ensure general consensus.

To try and achieve gold annotations, the annotation process was divided into two progressively more detailed assessments: A) the first phase focuses on the general comprehension of the message the translated text is trying to convey, following Unbabel's Annotation Guidelines, and also accounts for language specificities described in the proprietary Language Guidelines; B) the second and last phase is dedicated to customer support issues, such as translations related to greetings and closings. In other words, the main goal of this curation was to double-check if the annotations done by annotators were correct, necessary and appropriate.

Furthermore, every annotation done was associated with one of three different severity levels, *i.e. minor*, *major*, or *critical*, according to the Unbabel Annotations Guidelines. The severity level of an error depends on how much the accuracy, the fluency and/or the style of the translated text is affected by that error. In other words, the greater the negative impact an error has on the perceived quality of the text, the higher is the severity attributed to it. For a definition of each severity level, please refer to *Section 2.2.1.1.2*.

Note that the content for *Tables 4*, *5*, *6*, *7*, and *8* was retrieved from Unbabel's Annotation Guidelines (from typology version 2 in use until June 2022) when reviewing the annotations.

A. General Assessment

The first phase consisted of reviewing annotations and ensuring that they were correctly attributed to one of four major categories, namely *Accuracy*, *Fluency*, *Style*, and *Localization*. Note that for every given example, the source language is always English.

- ***Accuracy***

Annotations are classified as having an *accuracy* issue when the target text does not reflect the meaning of the source. For example, in instances where units are added and/or omitted in relation to the source text, as shown in the examples:

Addition - DE (target):

(1) Source: Hey Hannah,

Target: Hallo Hannah,**:)**

Omission - PT (target):

(2) Source: button.

Target: **[botão]**.

Moreover, *Accuracy* annotations also account for cases of *Untranslated* content that should have been translated and *MT Hallucinations*, where the content of the target text does not match with the source's content because of a disturbance in the MT system itself.

Untranslated - KO (target):

(3) Source: Hi There,

Target: 안녕하세요**There**,

MT Hallucinations - PT-BR (target):

(4) Source: Happy Playing!

Target: **Atenciosamente,**

Lastly, mistranslations such as the ones shown in *Table 4* fit within this category of *Accuracy* and significantly impair the comprehension of the translated text, because the target content does not represent the source.

Mistranslation Error Type	When to apply	Examples		
		TL	Source	Target
Named Entity	When names, places, locations and other named entities do not match between the source and the target	PT-BR	"**Chok**"	"**Escolha**"
	When any other type of error (e.g. *capitalization*) falls upon a named entity	TR	"Press the **PrtScn** key"	"Prtscn tuşuna bas"
False Friend	When the translation has a graphically similar word, but the meaning is different from the source word	NL	"Our customers and their experiences are really valuable to us and our **motive** is to provide a delightful customer service experience to them in every possible way."	"Onze klanten en hun ervaringen zijn erg waardevol voor ons en onze **motief** is om ze op elke mogelijke manier een geweldige klantenservice-ervaring te bieden."
Overly Literal	When the translation is too literal	EL	"**Here is** the account I'm seeing:"	"**Εδώ είναι** ο λογαριασμός που βλέπω:"
Lexical Selection	The target word does not convey the correct meaning of the source	SV	"They should then be able to begin an investigation with the aim of **retrieving** your funds."	"De bör då kunna påbörja en undersökning i syfte att **hämta** dina pengar."
Shouldn't Have Been Translated	The translated text should have been left untranslated	FI	"- **Serial number** (S/N) (Directly from the device, not from the product packaging):"	"- **sarjanumero** (sarjanumero) (suoraan laitteesta, ei tuotteen pakkauksesta):"
Source/Target Disagreement	There are either gender and/or number mismatches between the source and target	NO	"Proper **Cleaning Instructions**:"	"Riktig **rengjøringsinstruksjon**:"

Table 4. Accuracy - Segment annotation for different types of mistranslation (*Note: "TL" stands for Target Language)*

- ***Fluency***

Translations with *Fluency* issues are not well suited to the target language and may lead to an unnatural reading experience, therefore jeopardizing the text's readability and comprehension. This major category holds four other subcategories, namely: *Duplication*, *Inconsistency*, *Grammar*, and *Typography*.

Annotations must be classified as *Duplication* when, as the name suggests, a word or a bigger portion of the text is repeated unintentionally.

Duplication - TR (target):

(5) Source: S/N (Serial Number): (Under the left outer earplate)

Target: S/N (seri numarası) **(seri numarası)**: (Sol kulak plakasının altında)

When a target text has inconsistencies regarding the translation of the same terms or, in other words, when the same word is translated differently throughout the same text, the inconsistent segment must be annotated as *Inconsistency*.

Inconsistency - CS (target), where both examples belonged to the same translated text:

(6a) Source: **Refund in URL-0 **Credit****

Target: Vrácení peněz v URL-0 **kreditech**.

(6b) Source: We'll cancel your booking and instantly refund you with URL-0 **Credit**.

Target: Zrušíme vaši rezervaci a okamžitě vám vrátíme peníze v **kreditu** URL-0.

The second broadest subcategory within *Fluency* is related to grammar and syntax issues and is, therefore, called *Grammar*. This subcategory can be divided into three different types of classifications, as described in *Table 5*, *i.e. Function Words*, *Word Form*, and *Word Order*.

Grammar Error Type		When to apply
Function Words	Wrong Preposition	When a [function word] is used incorrectly and another [function word of the same category] should have been used instead
	Wrong Conjunction	
	Wrong Determiner	
	Wrong Pronoun	
	Wrong Auxiliary Verb	
Word Form	Agreement	When two or more words do not agree, regarding number, gender, person or case (if applied)
	Tense/Mood/Aspect	When a verbal form displays the wrong tense, mood, or aspect
	Part of Speech	When the part of speech is wrong, *i.e.* when a word is wrongly classified as an adjective, an adverb, a conjunction, a noun, an interjection, a preposition, a pronoun, or a verb
Word Order		When the order of the words is incorrect in the target sentence

Table 5. Fluency - Grammar subcategories and criteria for annotating segments

Finally, the last category related to *Fluency* accounts for issues related to the presentation of the text itself and is called *Typography*. This subcategory contains tags such as *Capitalization*, *Diacritics*, *Hyphenation*, *Orthography*, *Punctuation*, and *Whitespace*. These issue types will be explained further and examples will be provided for each one of them.

Capitalization - "Wrong use of capital letters or absence of capital letters."

(7) HU (target):
Source: Province:
Target: tartomány:

Diacritics - "Issues related to the use of diacritics (*i.e.* any mark placed over, under, or through a letter)". This tag must be applied when the diacritic is incorrect, missing or unnecessarily added to a letter.

ES (target):

(8) Source: We should be able to see signs of stick drift through the software.

Target: Deberíamos ser capaces de ver los signos de por qué **esta** suelta a través del software.

Hyphenation - Must use this tag when "the target text is hyphenated incorrectly, has a hyphen missing or has an extra hyphen"

FR (target):

(9) Source: Please respond directly from the email and reference this ticket number.

Target: Veuillez répondre directement par **email** et mentionner ce numéro du ticket.

Orthography - "Words spelled incorrectly. This category does not apply to errors related to diacritics or hyphens."

NL (target):

(10) Source: To help us get back to you faster, my response today is going via a translation tool.

Target: Om u sneller ondersteuning te bieden maken wij vandaag gebruik van een **vertaalsprogramma**.

Punctuation - This tag is used when the punctuation used is incorrect or missing (*e.g.* " ", ' ', ( ), [ ], { }, ¿ ?, or ¡ !).

PT-BR (target):

(11) Source: Hi João,

Target: Olá, João **[,]**

Whitespace - When a whitespace is incorrectly added or is missing.

SV (target):

(12) Source: **- Unpair**/repair or disconnect/reconnect hardware.

Target: **-Para** ifrån/para ihop eller koppla ifrån/koppla tillbaka hårdvaran.

- ***Style***

When the translated text has stylistic problems and/or issues related to addressing customers. There are three different categories the errors must fall in: *Grammatical Register*, *Lexical Register* and *Wrong Language Variety*. While the *Grammatical Register* subcategory must be used if "the text uses pronouns and verb forms that are not compliant with the register required", the *Lexical Register* tag is used when the same occurs with lexical expressions instead. For example:

Grammatical Register - AR (target): The translation should have been done following a formal register, but the pronoun used has an informal tone.

(13) Source: We look forward to hearing back from **you**.
Target: ننتظر أن أسمع **منك** قريباً

Lexical Register - ID (target): The translation should have been done following a formal register, but the expression has an informal tone.

(14) Source: Hi there
Target: **Hai**,

Regarding the *Wrong Language Variety* tag, a word/expression must be classified as such when the language variety is not the requested one. This tag is often associated with the following three language pairs: ES and ES-LATAM, PT and PT-BR, and ZH-TW and ZH-CN.

ES and ES-LATAM

(15) Source: It generally takes 5-10 business days for the payment processor to credit the **money** back to your account.
ES Target: (...) para que el procesador de pagos acredite el **dinero** a tu cuenta.
ES-LATAM Target: (...) para que el procesador de pagos acredite la **plata** a tu cuenta.

PT and PT-BR

(16) Source: The **username** on your original account
PT Target: O nome de **utilizador** (...)
PT-BR Target: O nome de **usuário** (...)

ZH-TW and ZH-CN

(17) Source: You can read more about account deactivation on our help center
ZH-TW Target: 我們正在努力盡快更新此信息。
ZH-CN Target: 我們正在努力盡快更新此資料。

- ***Localization***

The *Localization* category consists of five different annotation types that can be considered as locale conventions, such as *Numerals*, *Symbols*, *Currency, Date* and *Time Formats*. It is worth mentioning that localization and translation are two concepts that can lead to confusion. Although translation is indeed part of localization, multiple aspects are required for a term to be locally accepted. This is due to the fact that translations are related to the message the text is trying to convey, whereas localization transforms this message for a more tailored customer experience.

The *Numerals* category is annotated as *Wrong Number* and changes depending on the text's target language, as described in *Table 6*. This tag is used when there is an inconsistency between the source's numbers and the target's or when the translation does not follow the target language rules. Note that a language may allow more than one option in regards to the separator used, and therefore, it may appear twice in the following table. In addition, some Language Guidelines did not include information about this matter, which is the reason for their respective target language not being included in the table.

To indicate groups of thousands:	Target Languages	To indicate the decimal place:	Target Languages
Comma [,]	ES-LATAM / JA / KO / TH / ZH-TW	Comma [,]	DA / DE / EL / ES / FI / FR / HU / ID / IT / NO / PL / PT / PT-BR / RO / RU / SV / TR / VI
Period [.]	AR / DA / DE / EL / HU / ID / IT / NL / NO / RO / TR / VI	Period [.]	ES / ES-LATAM / JA / TH / ZH-CN
Whitespace [ ]	CS / ES / FI / FR / HU / KO / NO / PL / PT / RO / RU / SV	Whitespace [ ]	-
No punctuation [ø]	ZH-CN	No punctuation [ø]	ZH-TW

Table 6. Localization - Numerals: Language specificities when indicating groups of thousands and decimal places

The second category mentioned, *Symbols*, does not have a direct correlation with a specific tag. Instead, the annotation is classified depending on the issue associated with the symbol in question. In other words, symbols such as "%", "‰", "&", "+", "-", "±", "/", "|", "( )", "{ }", "[ ]", "ºC" and "ºF" can be preceded or followed by numbers and/or whitespaces, depending on the target language. If the order between symbol and number is incorrect, then the expression must be tagged as *Word Order*, whereas if the problem is related to existing or missing whitespaces, then the annotation used must be *Whitespace* instead.

Moreover, translation issues related to currency belong to the category *Currency*, albeit not having an associated annotation type, similarly to the former category. If there is an error involving the substitution of currency symbol conversions, the currency must be tagged as *Wrong Unit Conversion*. However, this tag can also be applied to other unit conversion issues, such as the one given in the example below, related to metrics.

Wrong Unit Conversion - VI (target):

(18) Source: If your position is just a little inaccurate (less than 100 **inches**) we recommend you to enable WiFi on your device.

Target: Nếu vị trí của bạn chỉ không chính xác một chút (dưới 100 **mét**), chúng tôi khuyên bạn nên bật WiFi trên thiết.

The last two localization categories concerning formatting issues are *Date Format* and *Time Format*. Translations where either dates or times do not match with the source text, must be annotated as *Wrong Date/Time*. The former can be influenced by either one or two variables: the separator used for distinguishing days, months and years, and the order in which these numbers are presented. *Table 7* covers all possibilities for the target languages Unbabel supports.

Date Order	Date Separator	Target Languages
Year Month Day	yyyy/mm/dd	AR / JA
	yyyy. mm. dd	HU / KO
	yyyy-mm-dd	SV
	YYYY 年 MM 月 DD 日	ZH-TW / ZH-CN
Day Month Year	dd.mm.yyyy	DA / DE / ES / FI / PL / RU
	dd. mm. yyyy	CS / NO
	dd/mm/yy	EL / ES / ES-LATAM / FR / HI / IT / PT / PT-BR / RO / TH / TR / VI
	dd-mm-yyyy	ES / ID / NL
	dd mm yyyy	ID

Table 7. Localization - Date Format: Language specificities when referring to dates

Lastly, the *Time Format* followed depends once more on the target language the text is being translated into. Some languages require that the time must be stated in a 24-hour format, while others require a 12-hour format. However, there is also the possibility of using both, for

languages such as ES-LATAM and JA, with the condition of maintaining one format throughout the same text to ensure consistency throughout the translation.

- **Language-specific Assessment**

The last step in the annotation curation within the General Assessment accounts for language specificities described in the proprietary Language Guidelines.

For each target language, Unbabel's guidelines consist of rules related to grammar, namely agreement issues with verbs, nouns and adjectives, and the use of determiners, prepositions and pronouns. For instance, the sentence's verb in languages such as French, Italian, or Portuguese must always be governed by the subject, and the adjective agrees in gender and number with the related noun or pronoun. If in a given translation for such languages those grammar rules are not met, then the incorrect tokens must be annotated accordingly.

B. Assessment on Customer Support rules

The Customer Support guidelines that will now be described apply only to *tickets*, *i.e.* emails from customers. They gather some of the previously mentioned rules above and refine them to fit customer support's needs and, thereby, guarantee a more fluent interaction between the client and the brand. These rules are related to a context focused on greetings and closings and are created upon *Lexical Register*, *Punctuation*, and *Capitalization* rules.

- ***Lexical Register***

A translation's register reveals how brands address their customers and the type of relationship they have. Register may vary depending on multiple factors such as the company, the brand's image, the service being offered, the customers, and the target language. Therefore, it is important to be familiar with the different lexical choices, adopt the correct register and adjust accordingly. With this in mind, Unbabel created a list of appropriate words and expressions to use in greetings and closings, depending on the register requested, ranging from a formal register

to a more colloquial one. For a list of selected expressions depending on the required register, please refer to *Table 1* in Annexes.

Some examples of this difference between formal and informal greetings and closings for Portuguese are given below.

Lexical Register - PT (target):

Greetings - Formal Register

(21a) Source: Hello Maria
Target: Cara Maria

Greetings - Informal Register

(21b) Source: Hello
Target: Olá

Closings - Formal Register

(21c) Source: Warmly
Target: Cordialmente

Closings - Informal Register

(21d) Source: Bye
Target: Até breve

All these concerns need to be taken into account, and thus, it is crucial for a brand to follow a detailed list of required expressions to promote positive first-impressions and to improve overall customer satisfaction.

- ***Punctuation***

When addressing customers, correcting punctuation errors in greetings and closings is as essential as following the just mentioned *Lexical Register* convention. In a customer service's context, each language requires a specific composition to greet and/or send off the addressee,

specifically made to ensure the best first impression possible. For example, in Polish, greetings must be followed by an exclamation mark or a comma (22a), unless there is a proper noun, and in that case there must be an additional comma between the greeting and the vocative (22b). On the other hand, closings either occur without any punctuation mark (23a) or with a comma (23b). There is however an exception with closing expressions such as "serdecznie pozdrawiam" or "pozdrawiam", in which an exclamation mark may be used (23c).

Punctuation - PL (target):

Greetings

(22a) Source: Hello again,
Target: Witam ponownie**,**

(22b) Source: Hi Zofia,
Target: Witaj**,** Zofia**,**

Closings

(23a) Source: Best regards,
Target: Z wyrazami szacunku

(23b) Source: Have a nice day,
Target: Miłego dnia**,**

(23c) Source: Regards,
Target: Pozdrawiam**!**

These rules are key when translating customer-service related content, for the reasons already mentioned and also due to the fact that the *Capitalization* criterion, may vary depending on the punctuation mark used in the translation.

- ***Capitalization***

The last step in the annotations' curation process in regards to customer support was to validate if the capitalization rules were being compliant with. Depending on the POS category or the preceding sentence's punctuation, a token must be capitalized or kept in lowercase. Usually

the first word of the sentence following the greeting is capitalized, however there are some exceptions. For instance, in Italian the first word after the greeting must always be kept in lowercase if the closing ends with a comma, in contrast to English (24). Similarly, the first sentence's word in German must be written in lowercase if the greeting ends with a comma, but with a period it must be capitalized (25). *Table 8* compiles the complete list of which languages require lowercase/uppercase letters after a greeting.

Capitalization - IT (target)

(24) Source: "Hello,
Thank you for contacting [ ]."
Target: "Salve,
grazie per aver contattato il supporto di [ ]."

Capitalization - DE (target)

(25) Source: "Hello,
Kindly find the invoice attached."
Target: "Hallo[**./,**]
[**B**itte/**b**itte] schau dir die Rechnung im Anhang an."

Lowercase	Uppercase	Condition		Not Applicable
IT	AR / DA / EL / ES / ES-LATAM / FI / FR / HI / HU / ID / NL / NO / PT / PT-BR / RO / RU / SV / TH / TR / VI	CS	The first word after the greeting must be written in lowercase, unless it is a proper noun.	JA / KO / ZH-CN / ZH-TW
		DE	If the greeting ends with a comma, the following word must be written in lowercase.	
		PL	If the greeting ends with a comma, the following word must be written in lowercase.	

Table 8. Capitalization for Customer Support: Required casing for sentences following greetings

4.3.3. EDF Totals: Target Languages and Registers

Once the annotations' revision was completed, the EDF was ready to be used as a test suite to evaluate Surfboard rules. The new EDF, coming to a total of almost thirty thousand segments - *Table 9* - for twenty eight different target languages - *Figure 19* -, now contained a well balanced and representative number of curated examples for formal and informal MT translations.

Formal Segments	13894
Informal Segments	13617
Grand Total	**27511**

Table 9. Evaluation Dataframe's Content: Grand total of formal and informal segments

Figure 19 illustrates the total number of segments for each target language, separated by registers. Although the data for the formal register in HI is significantly lower in number than for the informal register, the overwhelming majority of languages have a well balanced total of segments.

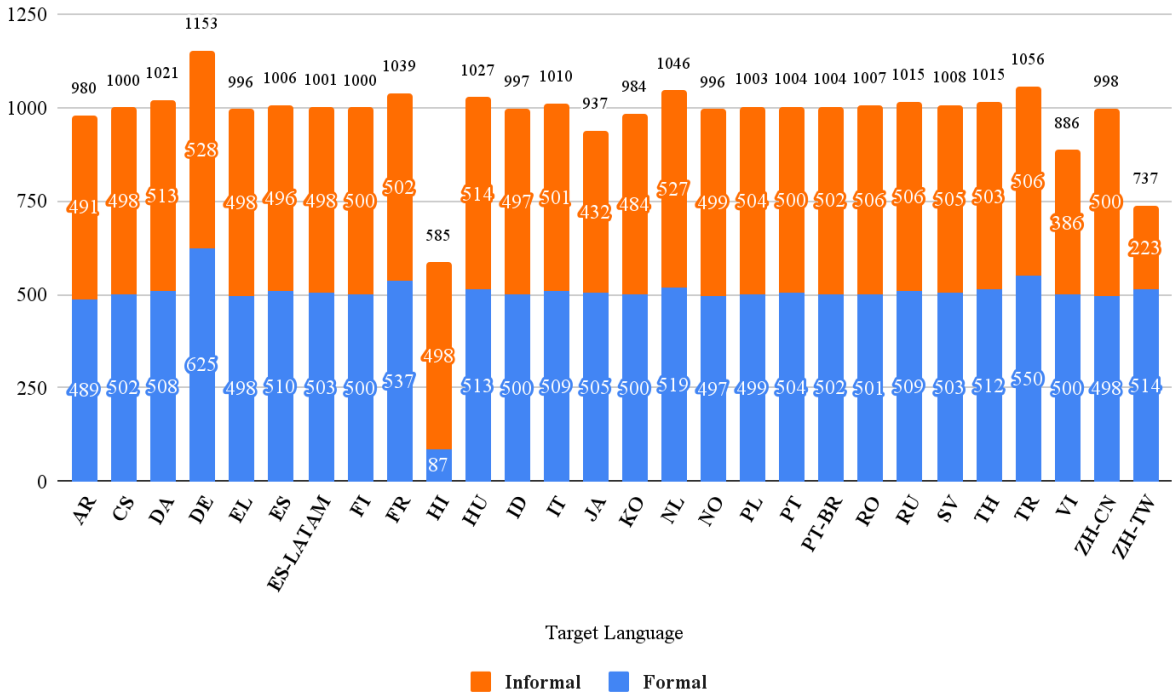


Figure 19. EDFs Content: Total of formal and informal segments per target language

In sum, the methodology presented throughout this section was established to analyze Smartcheck's performance. The first step consisted in a baseline analysis to allow for future comparisons. Once these results were gathered, we hypothesized that the reason for the low metric values obtained could lie in the evaluation dataframe being used to evaluate the system's performance, instead of the actual grammar-checking rules. Thereby, the initial dataframe (prior EDF) was analyzed, and it was concluded that the data was in need of being revised and updated. For this reason, new test suites were created in a systematic and methodical manner, with representative examples for the usual content the systems translate. These changes were made on the basis of detailed and thorough typologies in order to ensure the best possible quality.

The following section will present the results obtained after the implementation of the new test suites, *i.e.* new EDF, comparing baseline results and once again evaluating Smartcheck's performance.

5. Results and Discussion

The current chapter compares the impact of the new EDF with the previous one, and aims to analyze results obtained using the new curated data to evaluate different error detection systems. First, a comparison was made in *Section 5.1* regarding Surfboard rules in production comparing the results gathered from the baseline analysis in *Section 4.1* and the evaluation results gathered after the new EDF's implementation. Thereafter, Smartcheck was tested using the new and curated EDF as the gold standard in *Section 5.2*, in order to check its overall performance in detecting translation issues. This evaluation was done using a confusion matrix to measure this proprietary tool's performance, described in *Section 5.2.1*. The predicted annotations gathered from the confusion matrix were used to analyze the contrast between what Smartcheck considered as erroneous tokens/segments and which of those tokens/segments were indeed incorrect - *Section 5.2.1.1*. Once the results were gathered, cases of TPs, FNs, and FPs were used to calculate different accuracy measures for all target languages Unbabel supports in *Section 5.2.1.2*, namely Precision, Recall and F1-score. Following this comparative analysis between Smartcheck and gold annotations, *Section 5.3* describes the EDF's implementation in order to fulfill its purpose as a test suite and, therefore, evaluate other error detection systems, *i.e.* spell checkers. Finally, *Section 5.4* describes the importance of gathering reliable gold data to trust forthcoming rule's evaluations in order to monitor their quality and coverage, and assess whether there is a need for revisions or changes in any of them.

5.1. Rule's Evaluation Comparison between Baseline Analysis and new EDF

The results obtained in the baseline analysis were, as previously stated, not ideal for an error detection system. The elevated number of FN and FP cases show that Smartcheck was not yet capable of identifying most translation issues and, worse still, it detected problems where there were none. With this in mind, we hypothesized that the data being used to evaluate Smartcheck's rules might be one of the underlying causes of this problem. The previous EDF was, therefore, replaced by a new set of gold annotations, curated and revised by linguists for all language pairs supported by Unbabel. Thus, the first portion of the current section is dedicated to the rule's evaluation, after revamping the EDF, and to the comparison between the prior and the new results.

By the time the new EDF was implemented, it was important to check if the thirty nine rules previously evaluated were still enabled in production, since the baseline analysis had been done a few months prior to the new EDF's implementation.

After the verification was completed, Smartcheck was run through numerous new MT outputs from a given notebook and the metric values were gathered and compared. For a better understanding, take the following example into consideration: regarding the MT outputs used to evaluate rules in the baseline analysis, consider that there was an issue regarding a punctuation mark in greetings for the *X* target language; in order to automatically identify this language specific translation problem, Smartcheck retrieves Rule *A* (liable for punctuation demands in greetings for *X* target language) from Surfboard and gives a trigger warning. Consider that Rule *A* was one of the rules being evaluated in the baseline analysis. Following this, upon the new EDF's implementation, Smartcheck was run through new MT outputs and, once more, there was a punctuation issue for the same target language regarding greetings. In other words, both sets of MT outputs presented the same issues and, therefore, the same rule (Rule *A*) was triggered in both. If such conditions were met for other existing rules, then it would allow for a direct comparison of results between them. This is the case for the seven rules, presented in *Table 10*, out of the thirty nine previously evaluated. In other words, for an unambiguous comparison between rule's evaluation, solely those seven rules that were triggered in both evaluations will be compared.

It is also important to mention that the baseline comparison will focus solely on TP, FN, and FP cases.

		Prior EDF			New EDF		
Target Language	**Rule ID**	**TPs**	**FNs**	**FPs**	**TPs**	**FNs**	**FPs**
AR	117	No metrics			4	8	2
CS	125	No metrics			0	40	2
DA	115	No metrics			2	12	1
DE	12	0	0	899	0	21	1
ES-LATAM	28	0	16	1297	3	2	6
	110	3	9	1294	45	8	0
FR	45	59	420	750	123	67	1

Table 10. Rules' Comparison between baseline analysis and the new EDF

Once the results from both evaluations were placed against each other, three inferences were made: there is a notable difference regarding cases of FPs between the two evaluations; it is now possible to evaluate previously not-possible-to-evaluate rules; and, finally, the results obtained are significantly more reliable.

On one hand, the number of TPs, FNs and FPs were considerably different in both evaluations. FP cases in the baseline analysis were fairly high and, therefore, detrimental to the system's performance. However, the number of FPs in the second evaluation (post-new EDF's implementation) were significantly reduced. FN cases decreased as well, with the exception of rule 12 for DE. Moreover, the cases of TPs increased substantially for rules 110 and 45. As such, Surfboard rules (or at least those that were evaluated) do not have such low quality as previously stated. In other words, solely by changing the evaluation standard, the EDF, and with no alteration to the rules themselves, the rules Smartcheck relies on for error detection, although not perfect, are better than anticipated.

On the other hand, as was the case for the first three rules in *Table 10*, no results were available to assess if a rule had good or bad precision, coverage or overall value for a tool such as Smartcheck, since no metrics were obtained. Due to the new EDF however, every rule in Surfboard can now be evaluated. This was a key step in the quality assurance process, as MT systems can only improve if existing problems are correctly identified and the necessary changes are made. Knowing if a rule is not being triggered when it should be or if it considers good

quality translations as problematic is extremely important, because it allows us to know to a certain degree what to look for when revising rules.

Nevertheless, to draw any conclusions from the results obtained upon evaluation also means that the results are trustworthy enough to do so. Since the current EDF contains non duplicated data, representative examples that are up-to-date with Unbabel's content and annotations reviewed by linguists, it can be concluded that the new evaluation standard, and consequently its results, are proved to be much more reliable and accurate than the previous ones.

5.2. Testing Smartcheck with the new EDF

The following step was to analyze another correlation of interest regarding Smartcheck predictions. Without changing Surfboard rules, we decided to evaluate Smartcheck's annotations in comparison to the EDF's. In other words, one error detection system would be fed with the already existing Smartcheck's data, while a second error detection system would contain the EDF's reviewed data. Afterwards, each system would make predictions for the same translated segments, *i.e.* forecast which tokens would be detected by the system as having errors and annotate them accordingly. Therefore, our goal was to check how much this grammar-checking tool's annotations would differ in comparison to a new and curated standard. To that end, Smartcheck and the EDF were run through several translated segments and the results were gathered and separated into three parts: performance measurement with cases of TPs, FPs, and FNs; comparison between Smartcheck's and EDF's predicted annotations; and recalculation of metric values related to P, R, and F1-score.

5.2.1. Performance Measurement using a Confusion Matrix

Upon running Smartcheck and the EDF through translated segments, the values were gathered and, for purposes of facilitating this comparison, a confusion matrix[37] - *Figure 20* - was calculated to give a better overview of the results. The process of calculating a confusion matrix involves three different steps: i) acquiring a validation dataset with expected outcome values, *i.e.* the new EDF with the gold annotations (*y* axis called "True label"); ii) making predictions for

[37] **Confusion Matrix**: A technique used for summarizing the performance of a classification algorithm, in order to check what a classification model is capable of accounting for and what types of errors it is making.

each row in said dataset, or in our particular case, check Smartcheck's predictions - annotations - in comparison with the validation dataset (*x* axis called "Predicted Label"); and finally, iii) counting the number of correct and incorrect predictions for each annotation type.

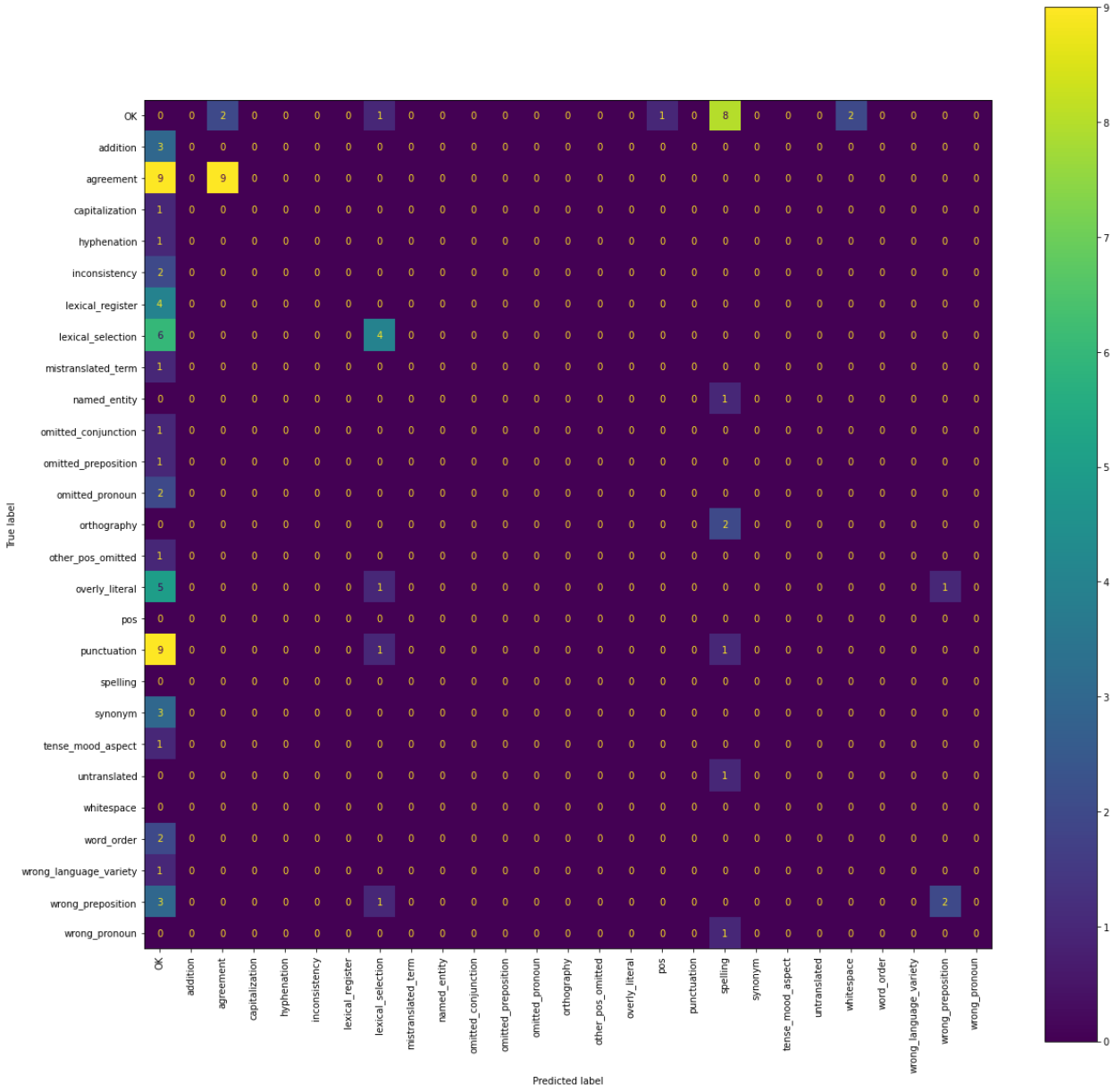


Figure 20. Confusion Matrix Example: Annotations for EN-ES (informal register)

Confusion matrices provide the number of FN and FP cases in the following manner: FN cases correspond to the first column "OK", while FPs correspond to the first row "OK".

Regarding TPs, these were divided into two groups: Labeled TPs, in which both the error span and the category were correctly detected according to the gold standard; and Unlabeled TPs in which the error span was correctly detected, but the category Smartcheck attributed the error to did not correspond to the "gold category". The intersection between labels of the same annotation type are instances of Labeled TPs, as is the case with the intersection between the row and column for "lexical_selection". However, regarding the row for "named_entity", its predicted label corresponds to a *Spelling* error annotation. In other words, Smartcheck correctly identified an error, but classified it incorrectly - Unlabeled TP. Moreover, confusion matrices also provide the number of gold instances of a label, *i.e.* the number of annotations Smartcheck would ideally predict regarding a specific error type. For that, the total number of FNs must be added to the total number of TPs.

Taking the confusion matrix's content into account, the total number of Smartcheck predictions came up to 4867 annotations, where FPs cases prevailed with 45.3%, FNs corresponded to 35.3%, and Unlabeled TPs summed up to almost 20%, as shown in *Figure 21*. Note that Labeled TPs, 8.3% of the total predictions, were accounted for within Unlabeled TPs, as the error span is correctly identified in both cases, solely differing in the error category. In other words, only 8.3% of Smartcheck's annotations fully matched with the EDF's.

By looking at the grand total of TPs, FNs and FPs in *Figure 21*, it is possible to conclude that many correct tokens were considered to be incorrect by Smartcheck - *i.e.* FP cases. High numbers of FPs are detrimental to any error detection system and must be avoided as much as possible, since it misleads us into believing in statistical evidence that is not real. Additionally, more than a third of all annotations were cases of FNs, thereby showing that the system overlooked a great number of existing translation errors. However, FN cases are not as damaging to a system's performance as FPs. This is due to the fact that we can create new or review already existing grammar checking rules in order to reduce said number of FN cases.

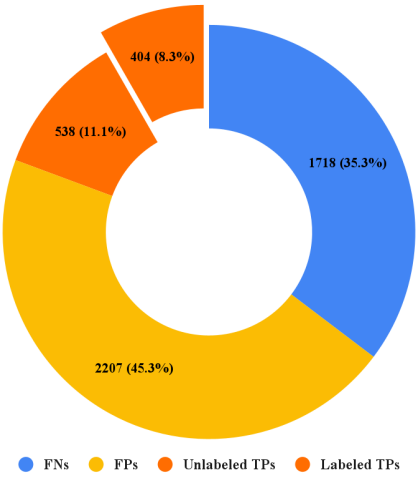


Figure 21. Grand Total of FNs, FPs and TPs when evaluating Smartcheck with the new EDF

5.2.1.1. Smartcheck *versus* EDF: Predicted Annotations Comparison

Upon the analysis of the different predictions, a comparison was made between every annotation done by Smartcheck and those done according to the EDF using the same data from the previous section. Note that for the current analysis we did not take into account specific target languages nor registers. In other words, *Figure 22*[38] shows the overall EDF's ideal annotation predictions in comparison to Smartcheck correctly detected annotations, *i.e.* labeled TP cases.

As illustrated in *Figure 22*, Smartcheck did not achieve the expected goal for any of the forty three EDF's detected error types. *Wrong language variety* annotations were the ones to reach closest to their target, with a difference of 5 tokens that Smartcheck did not account for. Other than that, the remaining Smartcheck annotations were far from achieving their target number, as it was only able to correctly detect and classify a reduced number of tokens for eleven error types: *Addition*, *Agreement*, *Lexical Selection*, *Orthography*, *Other POS Omitted*, *Overly Literal*, *Punctuation*, *Whitespace*, *Word Order*, *Wrong Preposition*, and *Untranslated*.

[38] **Error Types's Acronym Legend**: TMA stands for *Tense Mood Aspect*, TWA for *Term Wrongly Applied*, WDT for *Wrong Date Time*, SHBT *for Shouldn't Have Been Translated*, WLV for *Wrong Language Variety* and S.T.Disagreement for *Source Target Disagreement.*

Lastly, Smartcheck rules were not able to correctly predict any of the remaining thirty two error types.

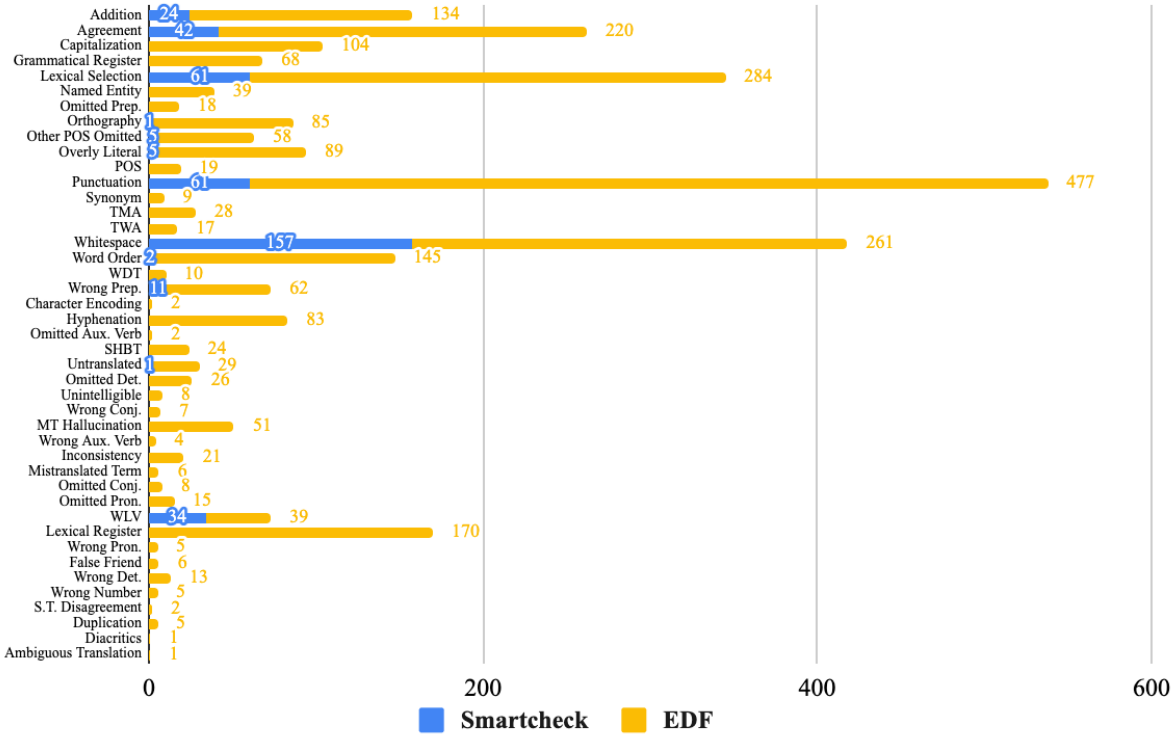


Figure 22. Smartcheck Labeled TP cases in comparison to gold annotations from the EDF

Analysis such as the one just described are extremely useful when testing an error detection system's performance, as it provides us with not only a general overview of the most problematic error types, but also with the possibility to isolate issues that would be otherwise concealed away.

5.2.1.2. Evaluating Smartcheck's Accuracy in Error Detection

The final step in the Smartcheck's testing process was to calculate the values for P, R and F1-score measures by taking into account the previous section's results, *i.e.* TP, FN and FP cases. Therefore, such values were calculated individually for every supported target language, focussing on the contrast between formal and informal registers. However, as sometimes there was necessary data missing (number of TP cases) not every language was possible to evaluate,

namely EL, HI, NL, NO, TR and VI. This is due to the fact that if a given language is missing its total number of TPs, then the values for P and R will not be possible to calculate and, consequently, neither will it be for F1-score.

In addition, some languages were excluded when calculating the average scores, namely CS and HU, and DA, ID, and PT, due to the fact that they only accounted for one of the supported registers (formal register regarding the former languages, and informal register regarding the latter) - as illustrated in *Figures 23*, *24*, and *25*.

- *Precision (P)*

To calculate the proportion of positive identifications done by Smartcheck that were actually correct, we must use P. Once calculated, the P value for the formal register was slightly higher on average in comparison to the average value for the informal register, as shown in *Table 11*.

Register	P Average
Formal	0.30354
Informal	0.30240

Table 11. P Average Total per Register

Regarding languages such as ES, ES-LATAM, IT, PT-BR, SV, and TH, the achieved values were notably higher for the informal register than for the formal register. However, this was not the case for languages such as FI, FR, JA, KO, PL, RU and ZH-TW in which the formal total was visibly higher.

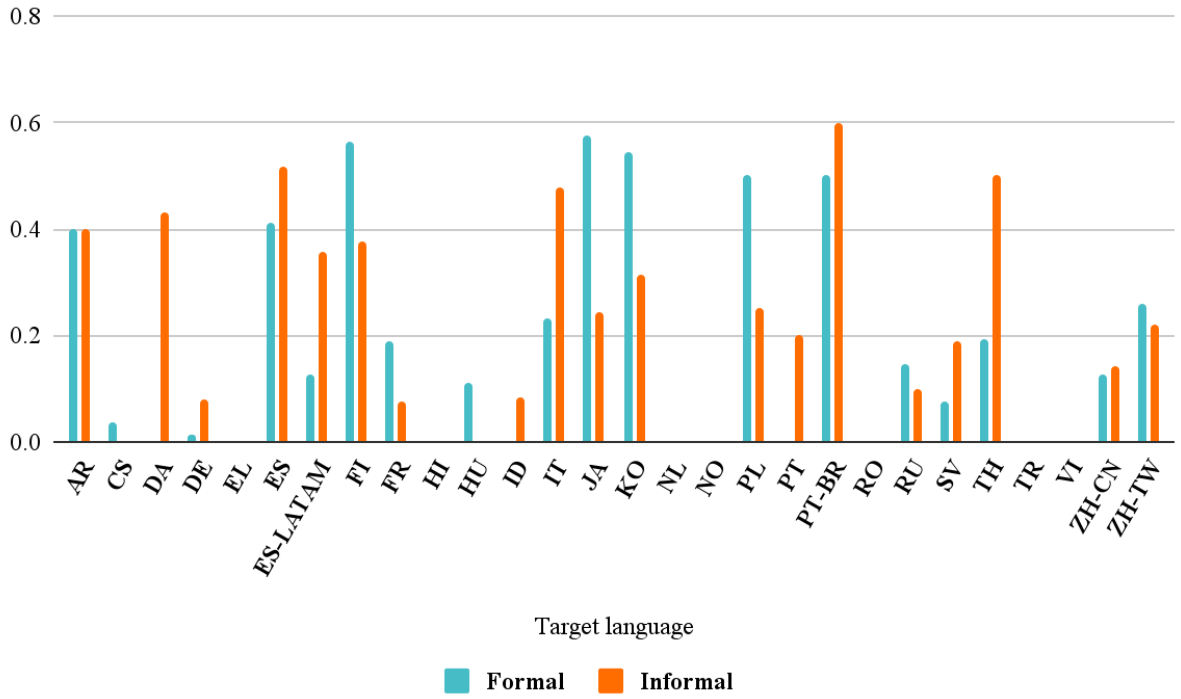


Figure 23. P Average per Target Language

- *Recall (R)*

The R measure aims to find the proportion of actual positives that were correctly identified by Smartcheck according to the new EDF. The R average is considerably higher for the formal register then for the informal one, as shown in *Table 12*.

Register	R Average
Formal	0.29626
Informal	0.24772

Table 12. R Average Total per Register

Similarly, but even more noticeable, the results obtained when calculating the P values are higher for formal registers (eleven languages in total) than for informal ones, as illustrated in

Figure 24. This divergence is clear in particular for languages such as FI, FR, JA, KO, PT-BR, SV, ZH-CN and ZH-TW.

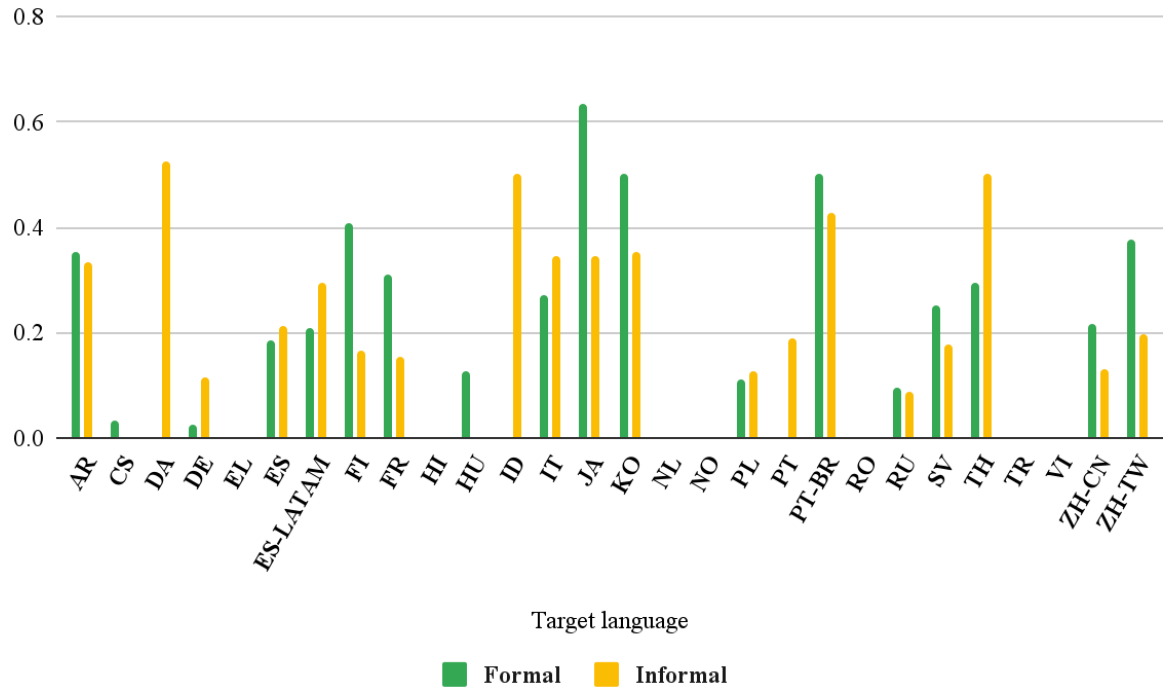


Figure 24. R Average per Target Language

- *F1-score*

The harmonic mean between P and R described above corresponds to the F1-score metric, used to statistically measure a systems' performance. As shown in *Table 13*, the formal register achieved a higher F1-score average in comparison to the informal register.

Register	F1-score Average
Formal	0.28141
Informal	0.26356

Table 13. F1-score Average Total per Register

For each supported language, the F1-score results were as illustrated in *Figure 25*. Languages such as FI, FR, JA, KO, ZH-CN, and ZH-TW consistently showed higher P and R values for the formal register, therefore their F1-score was also higher regarding that same register. However, for languages such as DE, ES, ES-LATAM, IT, and TH the same occurred for the informal register.

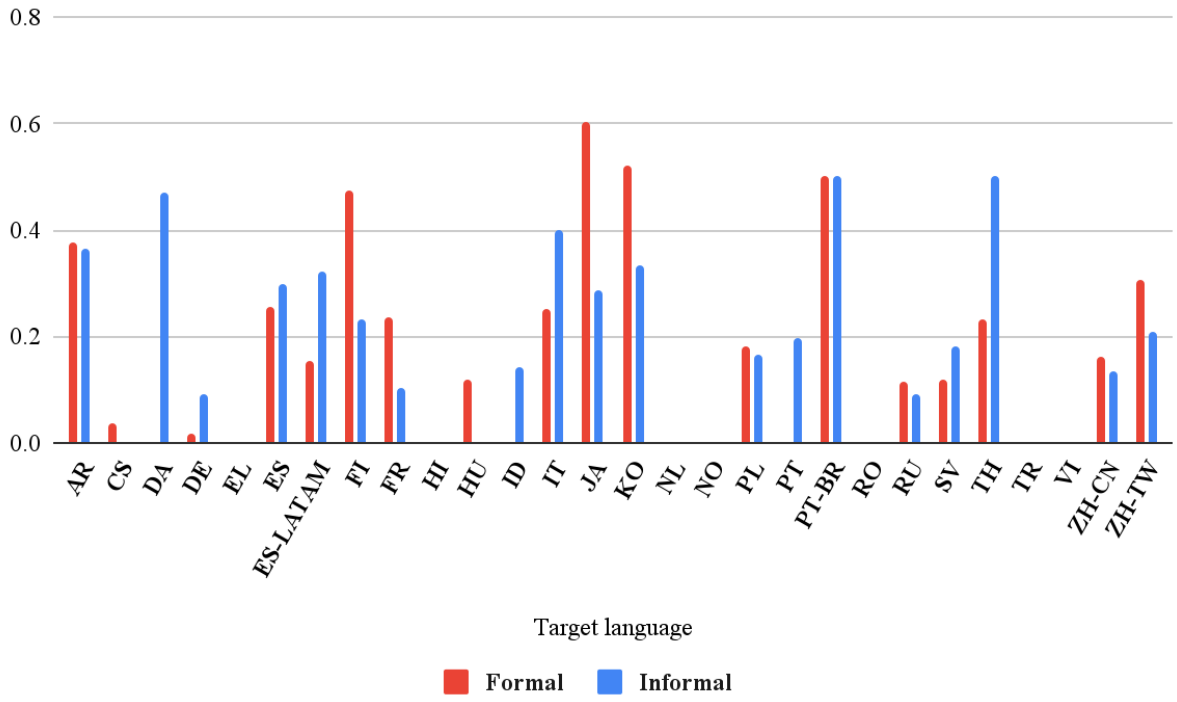


Figure 25. F1-score Average per Target Language

5.3. Quality Monitoring Assessment of Smartcheck Rules

Once the new EDF was implemented as the standard for system's evaluation, the results obtained regarding P and R measures provided great insights into the Smartcheck rules' value in detecting errors from MT outputs. However, as the F1-score metric consists of the mean between P and R, it does not take into account which of them is higher/lower, a crucial difference when assessing any rule's performance. For instance, consider Rule *A* to have 0.3 for P and 0.5 for R, while Rule *B* has 0.5 for P and 0.3 for R. Although the F1-score value is the same for both, each rule's error detection accuracy differs and, therefore, such contrasts must be accounted for. For

this reason, we propose an approach when evaluating grammar rules for monitoring error detection systems with reference to P and R values.

Consider only MT outputs that are revised by editors in post-edition to increase translation's quality. When a given rule is evaluated, the editor is provided with information regarding the rule's accuracy when detecting errors and, therefore, knows what to expect. In other words, when editors are faced with rules that achieve higher values for R in comparison to P, they know that, although noisy, that same rule will be able to detect the majority of instances of a specific error. Therefore, they must verify if the tokens or segments identified by that rule are indeed problematic and, if so, need to be corrected. On the other hand, if a rule achieves higher P values than for R, the editor knows that they can rely on it to correctly detect translation issues accounted for by that same rule and, therefore, there is not as much need to thoroughly revise the highlighted segments compared to rules with higher R values.

For a better understanding regarding the importance of P and R analysis, consider the following examples that show that it is much more problematic to have a system identify correct segments as incorrect than to overlook some minor translation errors. In other words, FNs are preferable to FP cases:

Example (1)

Rule *A*, created to identify critical errors, was evaluated as having a P value higher than R. In other words, the segments that are detected by Rule *A* are extremely likely to be indeed correct. It is also undeniable that overlooking critical errors leads to a great loss in the overall translations' quality. However, although Rule *A* is able to detect most of them, it cannot yet account for every single instance, due to its lower R value. Thus, Rule *A* must be adjusted in order to slightly increase R. Note that if we were to increase R to the point of surpassing P, then Rule *A* would trigger too many FP cases and, therefore, do more harm than good.

Example (2)

Rule *B* was specifically created to detect minor errors in MT outputs that do not have a strong impact in the fluency and comprehension of translated texts for a given language. However, once Rule *B* was evaluated, its value for R was much higher than for P. This means that Rule *B* considers many correct segments as incorrect and unnecessarily forces editors to

review correct translations. Therefore, increasing their time spent reviewing translations and, consequently, becoming a disadvantage for them. With this in mind, adjustments to the rule in question must be done to improve P and ideally decrease the number of FPs.

Thus, by accurately evaluating rules in error detection systems and analyzing the results, it is possible to obtain valuable information about the system's performance and make any necessary adjustments. This evaluation is only made possible with a reliable standard, *i.e.* the new and curated EDF.

5.4. Using the new EDF to Evaluate different Spell Checkers

As previously mentioned, the EDF contained reliable data regarding annotations of errors from MT outputs and, for that reason, it must be used as the gold standard to evaluate other error detection systems, *i.e.* fulfill its primary goal as a test suite. With this in mind, the gold data in the EDF was recently implemented and used to evaluate four different spell checkers used at Unbabel and, thereafter, choose which one would be the most appropriate to be deployed to production.

The evaluation was done similarly to the rule's assessment in the previous section, in which the number of TP, FP, and FN cases - illustrated in *Figures 26* - were obtained in order to calculate the values for P, R and F1-score - *Figure 27* - focusing on *orthography* annotations.

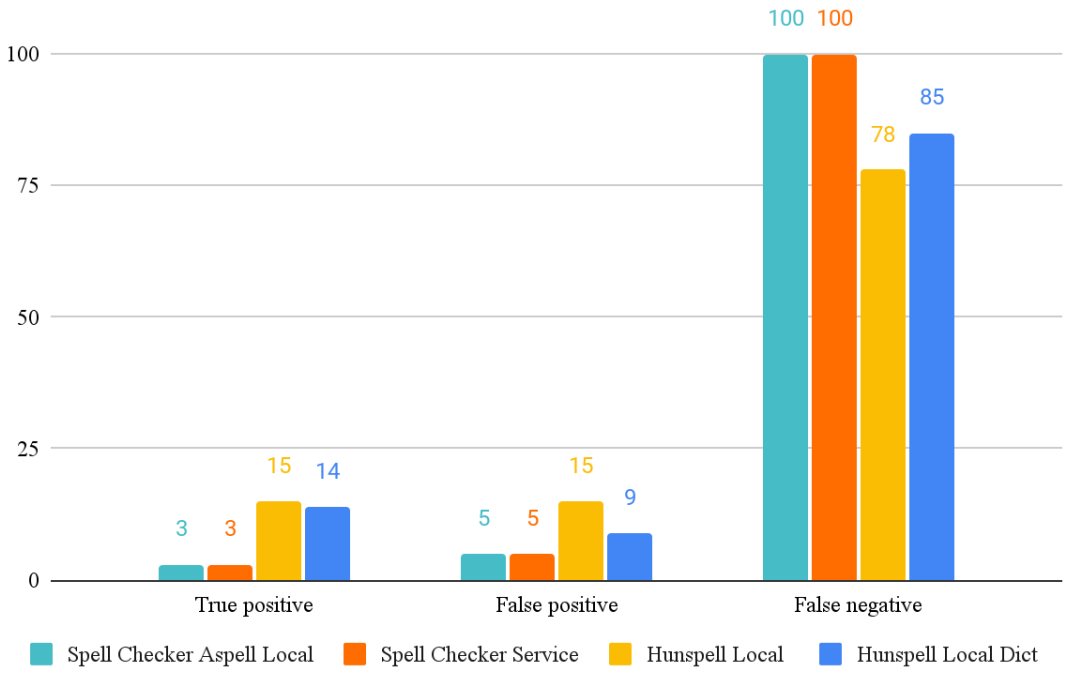


Figure 26. Spell Checkers Evaluation with new EDF as Gold Standard: Cases of TPs, FPs and FNs

Regarding the data in *Figure 26,* it can be seen that FN cases reached the highest numbers among the others, particularly regarding the first two spell checkers. Thus, this evaluation shows that a great number of existing translation errors (according to the EDF's data) are being overlooked by these spell checkers. In addition, the low number of TP cases demonstrates the lack of accuracy when identifying errors. In other words, using the EDF as the gold standard allowed us to conclude that most errors were not being detected and, therefore, it would be crucial to take corrective measures to recognize the root problem(s) in order to raise error detection's accuracy. This finding would not have been possible to reach had the standard being used to evaluate each spell checker not been revised and curated accordingly.

For the second evaluation done regarding the four spell checkers, the assessment focused on *orthography* errors contained in the EDF. As illustrated in *Figure 27*, the Hunspell Local Dict achieved the highest P value (almost 61%), while the Hunspell Local spell checker achieved a higher percentage of R and F1-score. Similarly to the previous evaluation, both the Spell

Checker Aspell Local and the Spell Checker Service demonstrated low equal results, thus showing once more a necessity for improvement.

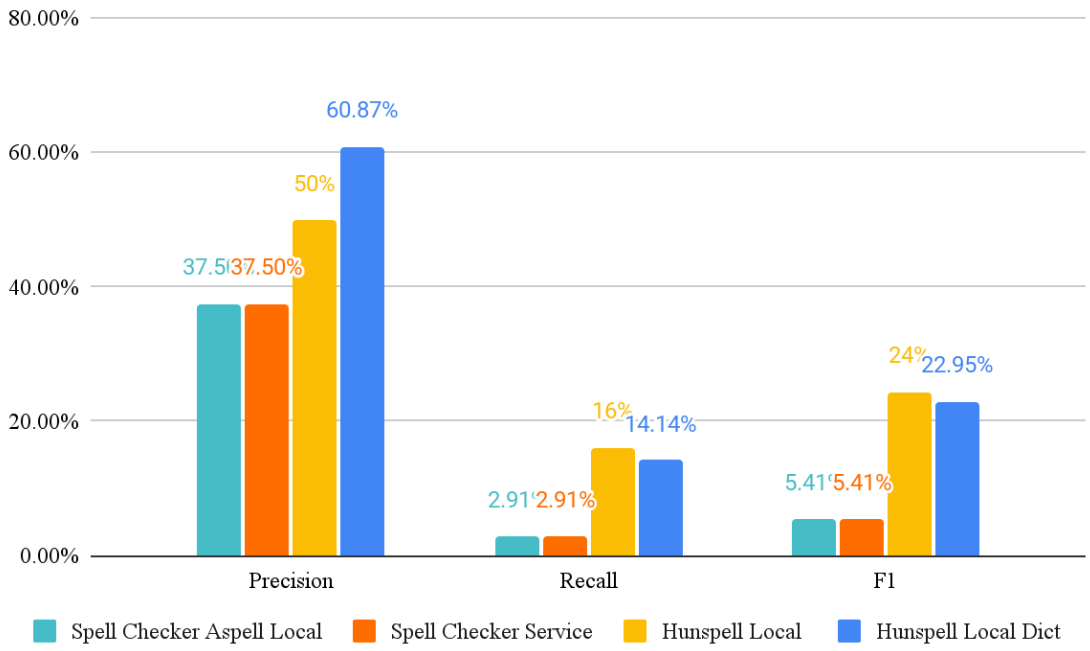


Figure 27. Spell Checkers Evaluation with new EDF as Gold Standard: P, R and F1-score

Both evaluations provided helpful insights regarding the performance of Unbabel's spell checkers, including the one currently in production - Aspell. However, one significant conclusion that was drawn upon evaluation was that both Hunspell spell checkers achieved much better results than the Aspell and the Spell Checker Service. For instance, the number of TPs for the Hunspell was higher, the total of FNs was lower, and the values for P, R and consequently for F1-score were considerably higher in comparison to the remainder. Upon this assessment, it was concluded that it would be beneficial to change the spell checker in production, from Aspell to Hunspell. Such a decision would not have been possible, had the EDF not been revised to provide truthful evaluations.

6. Conclusions and Future Work

The objectives of this report were threefold: i) to implement a methodology on creating reliable test suites for testing a proprietary tool on error detection and editing suggestions, named Smartcheck; ii) to evaluate the performance of this tool; iii) to contribute to the improvement on quality based on the edits suggested.

The work presented was implemented in several modules beyond the initial goal of creating test suites for the evaluation of Smartcheck. It was also applied to spell checkers evaluation and is now being applied also to measure the MT quality on the errors selected for the test suites developed within this thesis. Ultimately, the work conducted is also in production to evaluate in a suitable and consistent way the performance of each rule. As a future medium term project, we are now working on semi-automatic annotations based on the methodology conducted and the errors analyzed to provide assistance also to the annotators on semi-automatic suggestions of error annotations. Additionally, as a follow-up project we will also create a system to automatically check the viability of the rules by triggering different alerts, based on the annotations from the test suites.

We have started with a baseline analysis with the existing datasets and we then were able to verify that those datasets were not suitable nor reflected the core errors produced by the editors. Consequently, we have noticed that Smartcheck lacked accuracy when detecting and classifying errors. In an attempt to identify the root-cause of this system's limitation, we verified that the reason for the integrated rules not achieving ideal metric values might not be solely due to the rules' coverage and precision, but perhaps because the data being used to evaluate those rules is not suitable for its purpose. As such, this dissertation had the main goal of improving the methodology previously used when evaluating error detection systems, with particular attention to Smartcheck, in order to draw reliable conclusions about grammar-checking rules' coverage and accuracy in detecting translation errors. Moreover, improving Smartcheck will also benefit editors by assisting them on the editing process, reducing the time to identify translation errors, thereby making each task more time efficient.

For an error detection system to correctly identify and classify erroneous segments, it is crucial to first gather good quality and representative data for the content being dealt with, *i.e.* gold annotations. These annotations are then fed to systems such as Smartcheck and must be seen as their ideal outcome and used as the standard for evaluation, *i.e.* used as test suites. In

other words, Smartcheck heavily depends on annotated segments and the error typology used to classify them in order to provide editors with reliable suggestions. Thus, the test suites were created to be fine-grained selections of quality data.

With this in mind, the first step was to conduct a root-cause analysis of the existing evaluation set. This analysis led us to conclude that numerous annotations were incorrectly attributed, either due to the fact that there were no actual errors and the tokens were annotated regardless or because of improper error classification; the span of the errors were often not accurate and great number of associated severities were wrongly attributed to the annotations, in particular regarding the formal register. Finally, the existing set compiled an abundance of duplicated annotated segments, which is harmful to data evaluation corpora as it unnecessarily overloads them by not increasing its content representation.

Upon analyzing the existing dataset used to evaluate Smartcheck and its rules, it was found that there were indeed issues that needed to be addressed, data that needed to be filtered, and annotations that were in need of revision. Thus, instead of changing incorrect annotations and severities, adding missing annotations to overlooked errors, and removing duplicates from the test suites it was decided that it would be far more productive to gather up-to-date translations recently done at Unbabel, including content from Lingo24. In this manner, it was possible to compile suitable non duplicated data whilst balancing the number of segments between every language pair Unbabel supports and account for both registers (formal and informal). The methodological process of creating the new EDF is thoroughly described and explained in *Section 4.3*. In this manner, not only did it become possible to account for context dependent annotations with the new EDF, but the annotation curation process (in *Section 4.3.2*) that was followed assured that the data was trustworthy enough that any conclusions drawn from future evaluations would be valid and accurate, as was shown in our results and the fact that now the test suites created are in production for testing several modules of the pipeline.

The new evaluation standard allowed for every rule on Smartcheck to be possible to evaluate (a limitation of the previous evaluation set) and to conclude that the results obtained in the baseline analysis were in fact dubious. This is due to the fact that the metric values obtained from the re-evaluation regarding the seven comparable rules were not as low as previously stated. By solely changing the evaluation standard and with no alteration to the rules themselves, the rules Smartcheck relies on for error detection were better than anticipated. Moreover,

Smartcheck's performance was evaluated using the new EDF and the results were gathered within a confusion matrix. Therefore, allowing for a comparison between Smartcheck's annotations and gold predictions from the EDF, as described in *Section 5.2*. The evaluation was done considering cases of TPs, FNs and FPs (in order to calculate Precision, Recall and F1-score), which enabled evaluators to know the error types that were considered to be more problematic, and which languages and/or registers the error detection system could properly cover, in order to make the necessary adjustments. This knowledge on problematic error types can also be useful to editors, because we can alert them to pay more attention to specific errors and even use the data to train them in the Evaluation Tool.

Beyond the scope of our initial work and due to the reliability of the test suites created, the new EDF was also used to evaluate other error detection systems at Unbabel - spell checkers - and, through benchmarking comparison, choose which one would benefit the translation's quality the most so it could be used in production. With a trustworthy evaluation standard, a comparative analysis between systems' performance was possible to be conducted. Such results were gathered and it was possible to conclude which spell checkers performed better and should therefore be selected for production, and which ones incorrectly detected the highest number of tokens and must for this reason be improved. With solid evaluation methods for error detection systems, it becomes possible to create quality monitoring tools based on the results obtained from comparative analyses (that use curated data, such as the one compiled on the EDF, as its standard).

All the work developed throughout the internship at Unbabel has been thoroughly described and its relevance has been particularly demonstrated in the last two chapters. If the data used for the system's evaluation were to not be updated or revised, the results obtained would not be neither authentic nor reliable. In other words, there needs to be a reliable evaluation standard in order to assure that the conclusions drawn upon assessments are valid. The current work was able to therefore contribute to replicability and visibility regarding the methodological process for creating and curating test suites. As such, this dissertation provided a structured methodology for creating test suites based on curated annotations when evaluating error detection systems, whilst Lingo24's integration with Unbabel was still in progress (since the test suites that were created included gold curated data from both Unbabel's translated content as

well as Lingo24's). Additionally, tools from Lingo24's framework were also integrated into Smartcheck.

Future work will tackle a project on semi-automatic annotations to support the community of annotators at Unbabel, by providing suitable annotated segments and the corresponding error and severity type as a suggestion. Additionally, the work presented here provides a starting point for gold translations. By using the new EDF's data, it would only be necessary to add the gold version of the translated text, since the error would already be properly identified and classified. Efforts are now being conducted to use the test suites created to evaluate QE and MQM modules. We believe our work can also be a contribution to the creation of an alert system associated with concrete metrics and performance on the reliability of rules and data annotated. We are aware that the test suites will need updates and data augmentation strategies have already been tackled to cope with the challenges of the dynamics of data generation. Our work is in the process of being applied to research projects at Unbabel, especially the critical errors reduction on the Center for Responsible AI within the scope of the Plan of Resilience.

Annexes

	Greetings		Closings	
	Formal	**Informal**	**Formal**	**Informal**
DE	“Sehr geehrte/r” “Guten Tag” “Frau/Herr” (preferred)	“Hallo” / “Hi” / “Hey” / “Liebe(r)”	“Mit freundlichen Grüßen”	“Viele Grüße” “Beste Grüße” “Schöne Grüße”
ES	"Buenos días" / "Buenas tardes" / "Buenas noches" / "Estimado(a)"	"Hola" "Buenos días"	"Un saludo" "Un cordial saludo Saludos" "Saludos cordiales"	"Hasta luego" "Hasta pronto"
FR	"Cher Monsieur" "Chère Madame" "Bonjour"	"Salut" "Coucou"	"Meilleures salutations" "Au revoir"	"Salut" "À bientôt"
IT	"Salve" "Salve [name]" "Gentile [signor/signore, signora, sig., sig.ra, sig.na, name]"	"Ciao" "Ciao [name]"	"Cordiali saluti" "Distinti saluti"	"Ciao" "Saluti" "Arrivederci"
PT	"Excelentíssimo(a)" "Caro Senhor" "Cara Senhora" "Caro(a) [name]"	"Olá"	"Com os melhores cumprimentos" "Com elevada estima" "Coloco-me à sua inteira disposição para quaisquer esclarecimentos" "Atenciosamente", "Atentamente" "Cordialmente" "Respeitosamente"	"Até breve" "Até à próxima" "Cumprimentos"
PT-BR	"Olá" "Caro/a" "Prezado/a"	"Oi" "Olá"	"Atenciosamente" "Cordialmente"	"Até logo" "Até breve"

Table 1. Lexical Register for Customer Support: Selected examples of required expressions for Greetings and Closings, for both formal and informal registers

Target Language	Register	FNs	FPs	Unlabeled TPs	Labeled TPs
AR	Formal	11	9	8	6
	Informal	4	3	2	2
CS	Formal	30	25	1	1
	Informal	9	4	1	-
DA	Formal	-	-	3	-
	Informal	31	45	51	34
DE	Formal	36	67	20	1
	Informal	31	47	30	4
EL	Formal	4	10	-	-
	Informal	5	4	11	-
ES	Formal	62	20	24	14
	Informal	56	14	25	15
ES-LATAM	Formal	23	42	16	6
	Informal	24	18	15	10
FI	Formal	13	7	12	9
	Informal	15	5	7	3
FR	Formal	49	94	46	22
	Informal	49	107	29	9
HI	Formal	11	10	5	-
	Informal	49	28	20	-
HU	Formal	7	8	1	1
	Informal	4	4	-	-
ID	Formal	-	11	1	1
	Informal	1	11	1	1
IT	Formal	16	20	8	6
	Informal	19	11	16	10
JA	Formal	11	14	19	19
	Informal	17	28	18	9
KO	Formal	19	16	26	19
	Informal	20	24	59	11

NL	Formal	30	245	29	-
	Informal	49	208	35	-
NO	Formal	-	-	-	-
	Informal	-	-	1	-
PL	Formal	8	1	4	1
	Informal	14	6	7	2
PT	Formal	45	65	4	-
	Informal	17	16	7	4
PT-BR	Formal	1	1	4	1
	Informal	4	2	6	3
RO	Formal	22	12	28	-
	Informal	1	3	-	-
RU	Formal	202	122	39	21
	Informal	157	134	28	15
SV	Formal	3	12	3	1
	Informal	14	13	11	3
TH	Formal	12	21	10	5
	Informal	4	4	4	4
TR	Formal	6	16	1	-
	Informal	6	14	3	-
VI	Formal	2	9	-	-
	Informal	-	-	-	-
ZH-CN	Formal	94	178	69	26
	Informal	208	188	63	31
ZH-TW	Formal	74	128	64	45
	Informal	119	103	47	29
TOTAL		1718	2207	942	404

Table 2. Testing Smartcheck with new EDF: Total cases of FNs, FPs and TPs per register and target languages

Bibliography

Ailani, S., Dalvi, A., & Siddavatam, I. (2019). Grammatical error correction (GEC): research approaches till now. In *International Journal of Computer Applications*, *178*(40), (pp. 1-3)

Avramidis, E., Macketanz, V., Strohriegel, U., & Uszkoreit, H. (2019). Linguistic Evaluation of German-English Machine Translation Using a Test Suite. In Proceedings of the Fourth Conference on Machine Translation (2) (pp. 445–454) https://doi.org/10.18653/v1/W19-5351

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. https://doi.org/10.48550/ARXIV.1409.0473

Balkan, L. (1994). Test Suites: some issues in their use and design. In Proceedings of the Second International Conference on Machine Translation: Ten years on

Balkan, L., Arnold, D., & Meije, S. (1994). Test suites for natural language processing. In Proceedings of Translating and the Computer 16

Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (pp. 65-72)

Bar-Hillel, Y. (1952). The present state of research on mechanical translation. In Proceedings of the Conference on Mechanical Translation

Bar-Hillel, Y. (1960). The Present Status of Automatic Translation of Languages. Adv. Comput. (1) (pp. 91-163)

Bell, S., Yannakoudakis, H., & Rei, M. (2019). Context is key: Grammatical error detection with contextual word representations. arXiv:1906.06593

Bhoi, B., Vyawahare, P., Avhad, P., & Patil, N. (2017). Data duplication avoidance in larger database. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (pp. 1-4)

Bodrumlu, T., Knight, K., & Ravi, S. (2009, June). A new objective function for word alignment. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing* (pp. 28-35)

Bota, L., Schneider, C., & Way, A. (2013). COACH: Designing a new CAT tool with Translator Interaction. In *Proceedings of Machine Translation Summit XIV: User track*.

Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (2019). The BEA-2019 Shared Task on Grammatical Error Correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 52–75) https://doi.org/10.18653/v1/W19-4406

Burchardt, A., Macketanz, V., Dehdari, J., Heigold, G., Jan-Thorsten, P., & Williams, P. (2017). A linguistic evaluation of rule-based, phrase-based, and neural MT engines. The Prague Bulletin of Mathematical Linguistics, 108(1)

Bustamante, F. R., & León, F. S. (1996). GramCheck: A grammar and style checker. *arXiv preprint cmp-lg/9607001*

Castilho, S., Doherty, S., Gaspari, F., & Moorkens, J. (2018). Approaches to human and machine translation quality assessment. In Translation quality assessment (pp. 9-38)

Castilho, S., Moorkens, J., Gaspari, F., Calixto, I., Tinsley, J., & Way, A. (2017). Is Neural Machine Translation the New State of the Art? *The Prague Bulletin of Mathematical Linguistics*, *108*(1), 109–120. https://doi.org/10.1515/pralin-2017-0013

Chatzikoumi, E. (2020). How to evaluate machine translation: A review of automated and human metrics. Natural Language Engineering, 26(2) (pp. 137-161)

Chiang, D. (2005, June). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)* (pp. 263-270)

Chollampatt, S., & Ng, H. T. (2018, April). A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI conference on artificial intelligence* 32(1)

Chollampatt, S., Wang, W., & Ng, H. T. (2019, July). Cross-sentence grammatical error correction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 435-445)

Choshen, L., & Abend, O. (2018). Automatic metric validation for grammatical error correction. arXiv:1804.11225

Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv:2003.10555*

Dale, R., Anisimoff, I., & Narroway, G. (2012, June). HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP* (pp. 54-62)

Dale, R., & Kilgarriff, A. (2011, September). Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation* (pp. 242-249)

Deek, F. P., & McHugh, J. A. M. (2007). Open Source: Technology and Policy (1st ed.). Cambridge University Press https://doi.org/10.1017/CBO9780511619526

DeNeefe, S., Knight, K., Wang, W., & Marcu, D. (2007, June). What can syntax-based mt learn from phrase-based mt?. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (pp. 755-763)

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North (pp. 4171–4186) https://doi.org/10.18653/v1/N19-1423

Elbeck, M., & Bacon, D. (2015). Toward Universal Definitions for Direct and Indirect Assessment. Journal of Education for Business, 90(5) (pp. 278–283) https://doi.org/10.1080/08832323.2015.1034064

Felice, M., & Briscoe, T. (2015). Towards a standard evaluation method for grammatical error detection and correction. In *Proceedings of the 2015 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 578-587)

Flickinger, D., Nerbonne, J., Sag, I.A. and Wasow, T., (1987). Toward Evaluation of NLP Systems. Hewlett-Packard Laboratories. In the 24th Annual Meeting of the Association for Computational Linguistics (ACL), Stanford

Görög, A. (2014). Quality evaluation today: the dynamic quality framework. In Proceedings of Translating and the Computer 36.

Grundkiewicz, R., & Junczys-Dowmunt, M. (2018). Near human-level performance in grammatical error correction with hybrid machine translation. arXiv:1804.05945

Guillou, L., & Hardmeier, C. (2016). Protest: A test suite for evaluating pronouns in machine translation. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 636-643)

Guidelines for the implementation of quality assurance frameworks for international and supranational organisations compiling statistics November (2009). In *Committee for the Coordination of Statistical Activities* (CCSA)

Han, N. R., Chodorow, M., & Leacock, C. (2006). Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, *12*(2), (pp. 115-129)

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), (pp. 1735-1780)

House, J. (2014). Translation quality assessment: Past and present. In Translation: A multidisciplinary approach (pp. 241-264)

Hutchins, W. J. (1995). Machine translation: A brief history. In Concise history of the language sciences (pp. 431-445)

Hutchins, W. J. (2001). Machine translation over fifty years. In Histoire epistémologie langage, 23(1) (pp. 7-31)

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., Martins, A. F. T., & Birch, A. (2018). *Marian: Fast Neural Machine Translation in C++*. https://doi.org/10.48550/ARXIV.1804.00344

Kaneko, M., Sakaizawa, Y., & Komachi, M. (2017, November). Grammatical error detection using error-and grammaticality-specific word embeddings. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing* (1) (pp. 40-48)

Kantor, Y., Katz, Y., Choshen, L., Cohen-Karlik, E., Liberman, N., Toledo, A., Menczel, A. & Slonim, N. (2019). Learning to combine grammatical error corrections. arXiv:1906.03897

Kasewa, S., Stenetorp, P., & Riedel, S. (2018). Wronging a right: Generating better errors to improve grammatical error detection. arXiv:1810.00668

Kenny, D. (2019) The Routledge Handbook of Translation and Philosophy. Routledge (pp. 428-445)

Kepler, F., Trénous, J., Treviso, M., Vera, M., & Martins, A. F. T. (2019). OpenKiwi: An Open Source Framework for Quality Estimation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (pp. 117–122) https://doi.org/10.18653/v1/P19-3020

King, M., & Falkedal, K. (1990). Using test suites in evaluation of machine translation systems. In COLING 1990: Papers presented to the 13th International Conference on Computational Linguistics (2)

Koehn, P. (2009). *Statistical Machine Translation* (1st ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511815829

Koehn, P. (2020). Neural Machine Translation (1st ed.). Cambridge University Press. https://doi.org/10.1017/9781108608480

Koehn, P., Och, F. J., & Marcu, D. (2003). *Statistical phrase-based translation*. University of Southern California Marina Del Rey Information Sciences Inst.

Lavie, A., & Denkowski, M. J. (2009). The Meteor metric for automatic evaluation of machine translation. In Machine Translation, 23(2–3) (pp. 105–115) https://doi.org/10.1007/s10590-009-9059-4

Leacock, C., Chodorow, M., Gamon, M., & Tetreault, J. (2014). Automated grammatical error detection for language learners. In *Synthesis lectures on human language technologies* *7*(1) (pp. 1-170)

Legrand, J., Auli, M., & Collobert, R. (2016). Neural network-based word alignment through score aggregation. arXiv:1606.09560

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692

Lommel, A. (2018). Metrics for translation quality assessment: a case for standardising error typologies. In Translation Quality Assessment (pp. 109-127) Springer, Cham

Lommel, A., Uszkoreit, H., & Burchardt, A. (2014). Multidimensional Quality Metrics (MQM): A Framework for Declaring and Describing Translation Quality Metrics. In Tradumàtica: Tecnologies de La Traducció (12) https://doi.org/10.5565/rev/tradumatica.77

Macketanz, V., Avramidis, E., Burchardt, A., & Uszkoreit, H. (2019). Fine-grained evaluation of German-English machine translation based on a test suite. arXiv:1910.07460

Makhoul, J., Kubala, F., Schwartz, R., & Weischedel, R. (1999, February). Performance measures for information extraction. In Proceedings of DARPA broadcast news workshop, Herndon, VA (249)

McCord, M. C. (1985). LMT: a Prolog-based Machine Translation system. In Nirenburg (1985) (pp. 179-182)

Melamed, I. D., Green, R., & Turian, J. (2003). Precision and recall of machine translation. In *Companion Volume of the Proceedings of HLT-NAACL 2003-Short Papers* (pp. 61-63)

Mizumoto, T., Hayashibe, Y., Komachi, M., Nagata, M., & Matsumoto, Y. (2012, December). The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters* (pp. 863-872)

Naber, D. (2003). A rule-based style and grammar checker.

Nerbonne, J., Netter, K., Kader Diagne, A., Klein, J., & Dickman, L. (1992). A Diagnostic Tool for German Syntax. Report DFKI D-92-03, Saarbr ucken. Also in: Neal, J. and Walter, S. eds. (1991). Natural Language Processing Systems Evaluation Workshop, Berkeley, Rome Laboratory. Report RL-TR-91-362, New York

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., & Bryant, C. (2014, June). The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task* (pp. 1-14)

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C., & Tetreault, J. (2013). The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2013 Shared Task*). Sofia, Bulgaria

Okpor, M. D. (2014). Machine translation approaches: issues and challenges. In International Journal of Computer Science Issues (IJCSI), 11(5) (pp. 159-165)

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the*

Association for Computational Linguistics (pp. 311-318) https://doi.org/10.3115/1073083.1073135

Rawling, P., & Wilson, P. (Eds.) (2019). The Routledge Handbook of Translation and Philosophy. London: Routledge.

Regnier, S. and Dauphin, E. (1994). Test Suite Design at Aerospatiale, In Input Paper for WP2, TSNLP

Rei, M., & Yannakoudakis, H. (2016). Compositional sequence labeling models for error detection in learner writing. arXiv:1607.06153

Rei, R., Farinha, A. C., de Souza, J. G., Ramos, P. G., Martins, A. F., Coheur, L., & Lavie, A. (2022). Searching for COMETINHO: The Little Metric That Could. In Proceedings of the 23rd Annual Conference of the European Association for Machine Translation (pp. 61-70).

Rei, R., Farinha, A. C., Stewart, C., Coheur, L., & Lavie, A. (2021). MT-Telescope: An interactive platform for contrastive evaluation of MT systems. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, (pp. 73–80) https://doi.org/10.18653/v1/2021.acl-demo.9

Rei, R., Stewart, C., Farinha, A. C., & Lavie, A. (2020). COMET: A Neural Framework for MT Evaluation. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 2685–2702) https://doi.org/10.18653/v1/2020.emnlp-main.213

Rothe, S., Mallinson, J., Malmi, E., Krause, S., & Severyn, A. (2021). A simple recipe for multilingual grammatical error correction. arXiv:2106.03830

Rozovskaya, A., Chang, K. W., Sammons, M., & Roth, D. (2013, August). The University of Illinois system in the CoNLL-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task* (pp. 13-19)

Rozovskaya, A., & Roth, D. (2014). Building a state-of-the-art grammatical error correction system. *Transactions of the Association for Computational Linguistics* (2) (pp. 419-434)

Schäler, R. (2004). Language resources and localisation. In *Proceedings of the Second International Workshop on Language Resources for Translation Work, Research and Training* (pp. 18-25)

Sellam, T., Das, D., & Parikh, A. P. (2020). BLEURT: Learning robust metrics for text generation. arXiv:2004.04696.

Simard, M., & Isabelle, P. (2009). Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of Machine Translation Summit XII: Papers*

Slocum, J. (1985). A survey of machine translation: Its history, current status and future prospects. In Computational linguistics, 11(1) (pp. 1-17)

Snover, M., Madnani, N., Dorr, B., & Schwartz, R. (2009). Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In Proceedings of the Fourth Workshop on Statistical Machine Translation (pp. 259-268).

Specia, L., Scarton, C., & Paetzold, G. H. (2018). *Quality Estimation for Machine Translation*. Springer International Publishing https://doi.org/10.1007/978-3-031-02168-8

Specia, L., Turchi, M., Cancedda, N., Cristianini, N., & Dymetman, M. (2009). Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th annual conference of the European association for machine translation*

Stewart, C., Gonçalves, M., Buchicchio, M., & Lavie, A. (2022). Business Critical Errors: A Framework for Adaptive Quality Feedback. In *Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas* (2) (pp. 231–256)

Tan, Z., Wang, S., Yang, Z., Chen, G., Huang, X., Sun, M., & Liu, Y. (2020). Neural machine translation: A review of methods, resources, and tools. In AI Open (1) (pp. 5-21)

Teng, C. M. (1999). Correcting Noisy Data. In *ICML* (99) (pp. 239-248)

Tetreault, J., & Chodorow, M. (2008, August). The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)* (pp. 865-872)

Tezcan, A., Hoste, V., & Macken, L. (2017). A neural network architecture for detecting grammatical errors in statistical machine translation. In *The Prague bulletin of mathematical linguistics* (108) (pp. 133-145)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. https://doi.org/10.48550/ARXIV.1706.03762

Wang, H., Wu, H., He, Z., Huang, L., & Ward Church, K. (2021). Progress in Machine Translation. In *Engineering* (pp. 2095-8099) https://doi.org/10.1016/j.eng.2021.03.023

Weaver, W. (1999). Warren Weaver Memorandum, July 1949. In MT News International, 22(5-6)

Yamada, K., & Knight, K. (2001, July). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (pp. 523-530)

Yuan, Z., Taslimipoor, S., Davis, C., & Bryant, C. (2021, November). Multi-class grammatical error detection for correction: A tale of two systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (pp. 8722-8736)

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. arXiv:1904.09675.

Zhao, Y., Jiang, N., Sun, W., & Wan, X. (2018, August). Overview of the nlpcc 2018 shared task: Grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing* (pp. 439-445). Springer, Cham