TÉCNICO LISBOA

ErrorIST: towards the automatic evaluation of editors

Tiago Manuel Ferrão dos Santos

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Dr. João de Almeida Varelas Graça

Examination Committee

Chairperson: Prof. João Emílio Segurado Pavão Martins
Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Prof. Bruno Emanuel da Graça Martins

November 2016

Dedicated to my parents, Lucinda and Manuel

Acknowledgments

Thanking people is a difficult endeavour, as I will never be able to list everyone who helped me in this battle. I shall, nonetheless, attempt it.

First of all, I would like to thank my parents, Lucinda and Manuel Santos, who had deep enough love and pockets to finance my search for knowledge. And of course, my brother, Diogo Santos, who jokingly mocked me all the way from Japan, unwittingly making sure I would persevere.

I'd also like to thank my wonderful thesis advisor Prof. Luísa Coheur who not only provided me with the necessary guidance to complete this thesis but tolerated my failures, supporting me when I was at my lowest like a caring mother.

Thanks to all of the wonderful people at Unbabel for giving me valuable input on my tool and also helping out with the evaluation. A special thank you to my advisor João Graça for permitting me to create ERRORIST, to Helena Moniz who always made time to help me, despite her busy schedule, to Nuno Santos, who helped create the sample for ERRORIST's evaluation, and Ramon Astudillo for helping me with the evaluation.

This would not have been possible without my friends, both inside and outside IST, amazing people who were forced to listen to my whining throughout an entire year. Starting with my "cellmates", Bruno Henriques, Nuno Xu, Rúben Rebelo, Vânia Mendonça, and Miguel Faria. A huge thanks to the Pires brothers, Miguel and André for always being there to provide friendly chat and advice. And of course, my dear friends, that while not directly there with me, provided me with much needed support, by either providing distraction or listening to my protests; in no particular order: Nuno Sousa, Joana Teixeira, Filipe Silva, Frederico Sabino, Cíntia Costa, Diogo Pedroso, Pedro Garcia, and Pedro Dias.

And last, and most importantly, the one who has most contributed to my mental health during this thesis and the one who was most neglected as I spent yet another night working on this; my better half, Inês Mendes. Your smile has been my pillar during all my struggles.

Resumo

Nesta tese de mestrado, apresentamos uma ferramenta configurável, ERRORIST, capaz de criar texto com erros e verificá-lo pelas suas correcções. ERRORIST é uma ferramenta extensível, capaz de inserir uma variedade de erros baseados numa taxonomia de erros orientada à tradução. ERRORIST foi desenvolvido com a Unbabel, uma start-up de tradução baseada em tradução automática auxiliada por edição crowd-source. A avaliação de editores é uma das suas preocupações, visto que eles usam editores crowd para corrigir texto traduzido automaticamente de forma a garantir qualidade. A inserção de erros artificais é configurável em relação à sua variedade. ERRORIST foca-se na inserção de erros artificiais em Português Europeu. Avaliamos esta ferramenta submetendo os seus erros para correcção por parte de editores e detectando as correcções automaticamente. Avaliar as correcções automaticamente é um desafio e apesar de ERRORIST não conseguir completamente substituir a avaliação manual das correcções, consegue criar material para as mesmas de forma fiável e rápida, ao mesmo tempo que reduz a necessidade de verificação manual das correcções. ERRORIST consegue reduzir a necessidade de verificações manuais em 66.38%, ainda mais (70.48%) se apenas considerarmos edição de documentos não traduzidos. ERRORIST provou a sua utilidade até num ambiente empresarial, criando erros adequados para avaliação de editores e recuperando 58.82% das correcções dos mesmos.

Palavras-chave: Avaliação, Erro, Erro artificial, Rastreabilidade

Abstract

In this master thesis, we present a configurable tool, ERRORIST, capable of creating error prone text and verifying it for corrections. ErrorIST is an extensible tool, capable of inserting a variety of errors based on a translation error taxonomy. ErrorIST was developed with Unbabel, a start-up based on crowd-sourced, machine aided translation. Editor evaluation is one of their concerns, since they use crowd editors to correct machine-translated text in order to ensure quality. Artificial error insertion is configurable on error type variety. ERRORIST focuses on the insertion of errors in European Portuguese. We evaluate this tool by submitting its errors to editor correction and detecting their corrections automatically. Evaluating the corrections accurately is a challenge and while ERRORIST cannot completely replace manual evaluation, it creates the evaluation material reliably and fast whilst reducing the need for human verification in correction detection. ERRORIST can reduce the need for manual correction verification in 66.83%, even further (70.48%) if you consider non-translation editing. Even in a business environment, ERRORIST proved its usefulness by creating adequate errors for editor evaluation and by tracing 58.82% of those errors' corrections.

Keywords: Artificial error, Error, Evaluation, Traceability

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This document describes an artificial error generation tool capable of introducing errors in European Portuguese (EP) sentences, ERRORIST. In this chapter we discuss the motivation behind the development of ERRORIST, the goals and main contributions of this thesis. Following this, we will present how the document is structured.

1.1 Motivation

This master's thesis was developed in the Spoken Language Systems Laboratory (L2F) in collaboration with Unbabel[1]. Unbabel is a start-up based on crowd-sourced, machine aided translation. In Unbabel, a translation job is first divided into multiple segments that are translated automatically, and, then, to ensure text quality, delegated to members of their community (non-professional editors) for editing (that is, detect and correct errors in translated text). These editors speak both the original language and the target language. In order to produce quality translations, Unbabel has to guarantee that their community editors are quality editors as well. This requires an evaluation method capable of grading the editors' work. One way to evaluate editors is to provide them error-prone text for correcting and use the original text to verify if the errors were identified and corrected.

Ideally, we can create these error-prone texts automatically by inserting errors in grammatically correct text, and, also automatically, verify the quality/coverage of corrections made after the editors' revision.

During editing, one of the concerns the editor (not necessarily just Unbabel's) should have, is the context. The context, as we understand, is any surrounding relevant information regarding any said text. Whether this information originates from other words in a sentence, its original non-translated form or client specifications, is not relevant to this definition.

Current error generation tools have other applications, like providing a way of creating an error annotated corpus for machine learning applications. In fact, most of the systems that use artificial error generation were made with the end goal of creating error annotated corpora for machine learning-based

[1]Unbabel: `https://unbabel.com/` Accessed last in 2016-10-10

error detection/correction systems; some of these were created with an even more specific focus on English learner errors, which are limited in variety. To the best of our knowledge, no current error generation system was created with the following in mind:

- Editor/Student linguistic evaluation
  Current artificial error insertion systems only focus on corpora generation for machine learning.
- Error Generation in European Portuguese
  Despite the existence of language neutral error insertion tools, EP specific error insertion is yet to be explored.
- Retrieving corrections for generated errors
  Generating errors is already possible, but verifying the sentences for corrections is not done by current error generation tools

Taking this into account, we propose a tool, ERRORIST, capable of inserting adequate human-like errors into text, in order to modify it in a way that will be used as a reliable resource for editor/student evaluation and machine learning training.

1.2 Objectives

We intend to create a general error insertion tool, which we will refer to as ERRORIST henceforth, with the following capabilities:

- Configurable Error Generation
  The ability to insert errors according to a desired frequency, language (EP or English) and difficulty.
- Traceable Error Generation
  The ability to insert errors that are, in some form, traceable to their origin.
- Correction Detection
  The ability do detect if an artificial error was corrected.
- Extensible tool
  The ability to extend the currently available tool to new languages, error types and error insertion methods.

Achieving these goals requires an in-depth analysis of existing error types, how to insert them, in which proportion and how to maintain or break grammatical correctness.

In order to evaluate the tool itself, we will use the following metrics:

- Error Quality
  A measure to represent if the insertions errors accurately represent their error type.

- Error traceability
  A measure representing how many of the generated errors were detected by the user and said corrections were indentified by the tool.

These two features will represent ERRORIST's performance in different ways. Error Quality represents the quality of each of the error types' insertion methods. Error traceability will measure how adequate ERRORIST is as an evaluation tool.

1.3 Thesis Outline

The document is organized as follows: in Chapter 2 we discuss the necessary knowledge and terms needed to read this work, including Section 2.4, in which we analyse the two error taxonomies that we will follow in this work plus two other relevant taxonomies, and Section 2.2, in which we discuss the state of the art regarding error insertion, available error generating tools and types of errors inserted by each of them; In Chapter 3 we describe the developed solution's architecture and its error insertion methods; In Chapter 4 we refer to the measures and procedures we used for evaluating ERRORIST and their respective results; Finally, in Chapter 5, we summarize this document and present some possibilities for ERRORIST's extension.

Chapter 2

Background

An error, as we define in our work, is any deviation from the language grammatical rules or intended meaning as defined at the time of writing. This includes slight meaning variations, as error severity is outside of this work's scope. **Artificial errors** are errors that were not created by accident at the time of writing, but errors that were inserted automatically by a computer. Artificial error insertion tools are not numerous, and those that exist were not created for the purpose of our work: editor evaluation. Firstly, we will enumerate some relevant error taxonomies that were taken into account in our work. Starting from the simplest taxonomy, to the most complex. Following that, we describe some systems that generate errors. We also discuss how error generation has contributed to machine learning applications and, finally, we will describe some resources useful for error insertion.

2.1 Error Taxonomies

There is more than once way to classify an error. Here, we present the taxonomies relevant in ERROR-IST's development.

For a better understanding of the errors we refer to, examples for pertinent errors will be provided throughout this document. These examples are written according to the following notation: brackets mark where a word was removed, '[ ]', parenthesis mark where a word was added, '(extra)', and underlining marks where the error was introduced, '<u>wrong</u>'.

2.1.1 Basic taxonomy

The first taxonomy to consider is the simplest one, as any error can be categorized as one of the following:

- Addition, in which a word is added.
- Omission, in which a word is removed.
- Substitution, in which a word is replaced by another.

- Move, in which a word is moved to another part of the sentence.

GenERRate[1], as explained in Section 2.2.1, used this error categorization for artificial error insertion.

2.1.2 L2F taxonomy

The second taxonomy we will consider is the one presented by Costa et al.[2] developed in L2F. In this work they tried to categorize machine translation errors in several levels. We will now analyze the error types they defined and see how these can occur. We will also provide examples for each error type when possible, in both English (EN) and EP.

It is important to note that even though we did not create this taxonomy, the error types have been adapted when possible to suit our tool's purpose, which is not solely focused on machine translation errors.

Orthography Level

The orthography level includes all errors related to misspelling and punctuation misuse. The error types included in this level and examples for each are in Table 2.1.

Error Type	Lang	Intended Sentence	Error Sentence
Punctuation	EN	I found the clowns, Bob, and Clyde.	I found, the clowns, Bob, and Clyde.
	EP	Encontrei os palhaços, João, e Cláudio.	Encontrei os, palhaços, João, e Cláudio.
Capitalization	EN	I think my poor Slipper got dirty!	i think my poor Slipper got dirty!
	EP	A minha pobre Pantufa ficou suja!	a minha pobre Pantufa ficou suja!
Spelling	EN	I have three friends.	I have htree friends.
	EP	Eu fui para a casa.	Eu fiu para a casa.

Table 2.1: Orthography level error examples

Punctuation errors relate to misused punctuation. **Capitalization** errors concern words that should begin with a capitalized letters, but are not, and vice-versa. **Spelling** errors, as the name indicates, relate to word misspellings.

Lexis Level

Lexis level errors include errors regarding the word as a whole and how its presence affects the sentence. The error types included in this level and examples for each are in Table 2.2.

According to Costa et al.[2], a **function word** is a word that has little lexical meaning but provides a way of expressing grammatical relationships with other words (e.g. he, of, it, already). **Content**

Error Type	Lang	Intended Sentence	Error Sentence
Omission (function word)	EN	On his birthday he tried a new hat on.	[ ] his birthday he tried a new hat on.
	EP	Ele já recebeu um chapéu no seu aniversário.	Ele já recebeu um chapéu [ ] seu aniversário.
Addition (function word)	EN	He bought a hat.	He bought a (already) hat.
	EP	Ele comprou um chapéu.	Ele comprou um (já) chapéu.
Omission (content word)	EN	His hat was prettier.	His hat was [ ].
	EP	O seu chapéu era bonito.	O seu chapéu era [ ].
Addition (content word)	EN	His was prettier.	(suit) His was prettier.
	EP	O seu era mais bonito.	(chapéu) O seu era mais bonito.
Unstranslated	From EN	Boys like bugs, as girls like dresses.	Boys like bugs, as girls like dresses.
	To EP	Os meninos gostam de insectos, como as meninas gostam de vestidos.	Os boys gostam de insectos, como as meninas gostam de vestidos.
	From EP	Os meninos gostam de insectos, as meninas gostam de vestidos.	Os meninos gostam de insectos, as meninas gostam de vestidos.
	To EN	Boys like bugs, girls like dresses.	Boys like insectos, girls like dresses.

Table 2.2: Lexis level error examples

words, according to Costa et al.[2], are words that carry the content or the meaning of the sentence (e.g. dog, house, happy, jump). **Omission** and **Addition** errors concern the omission and addition of content/function words respectively. **Untranslated** errors occur when words in a language should have been translated, but were not.

Grammar Level

Grammar level errors include errors regarding morphological or syntactic deviation from the language grammatical rules. The error types included in this level and examples for each are in Table 2.3.

Error Type	Lang	Intended Sentence	Error Sentence
Misselection (word class)	EN	The cute bird is happily on the branch.	The cutely bird is happily on the branch.
	EP	O lindo pássaro estava alegremente na árvore.	O lindamente pássaro estava alegremente na árvore.
Misselection (verb level: tense)	EN	He had bought a suitcase for his travels.	He had buy a suitcase for his travels.
	EP	Ele tinha comprado uma mala para as suas viagens.	Ele ter comprado uma mala para as suas viagens.
Misselection (verb level: person)	EN	This car is destroyed.	This car am destroyed.
	EP	Come, porque a viagem é longa!	Come, porque a viagem são longa!
Misselection (verb level: blend)	EN	They were beautiful yesterday.	They am beautiful yesterday.
	EP	Ele comeu enquanto pensava nas alturas em que jogaria à bola.	Ele comeriam enquanto pensava nas alturas em que jogaria à bola.
Misselection (agreement: gender)	EN	She tied her hair into a knot.	She tied his hair into a knot.
	EP	Ele atou os cabos do computador por ela.	Ele atou os cabos da computador por ela.
Misselection (agreement: number)	EN	The wolf took care of many cubs.	The wolf took care of many cub.
	EP	O lobo já os tinha alimentado.	Os lobo já os tinha alimentado.
Misselection (agreement: person)	EN	We learn from our mistakes.	We learn from my mistakes.
	EP	Aprendemos com os nossos erros.	Aprendemos com os meus errors.

Misselection (agreement: number)	EN	The wolf took care of many cubs.	The wolf took care of many cub.
	EP	O lobo já os tinha alimentado.	Os lobo já os tinha alimentado.
Misselection (agreement: blend)	EN	All hail our many huntresses in the hunter's guild!	All hail our many hunter in the hunter's guild!
	EP	Vou àquela loja de roupa para meninas.	Vou àqueles loja de roupa para meninas.
Misselection (contraction)	EN	N/A	N/A
	EP	Ela adorava sentar-se no banco.	Ela adorava sentar-se em o banco.
Misordering	EN	I like the beautiful colors on the car.	I beautiful like the [ ] colors on the car.

Table 2.3: Grammar level error examples

Misselection errors occur when a word was chosen incorrectly to convey the intended meaning but still has a semantic relation to the correct word. **Misselection (Word class)** errors are errors that use the the wrong class of a correct word, like a noun instead of an adjective.

Misselection (Verb level: Tense) errors are errors in which the verb tense was used incorrectly. In the example, the word in cause is a verb which tense was chosen incorrectly, this can create a invisible error as long as there is no other word in the sentence related to the tense.

Misselection (Verb level: Person) errors are errors in which the verb person was used incorrectly. We could not find any error example in the English language for invisible errors regarding the verb person due to the regularity exhibited between persons. In English, most verbs remain unchanged in the various persons, except in the third which makes the transformation into another person either the same word or a visible error. No example was found for this error in the English language, but if it exists, the example lies on homograph properties between two irregular verbs.

Misselection (Verb level: Blend) errors are a combination of the previous two. In the example provided, only the person is immediately noticeable as an error (if you do not take the semantics of the word 'yesterday' into account).

Misselection (Agreement: Gender) errors are errors in which two related words are not in the same gender. In the fifth example, the wrong gender of the possessive adjective 'his' is used instead of 'her' when referring to a girl. In the invisible example, changing the gender of the underlined words did not make the sentence incorrect grammatically.

Misselection (Agreement: Number) errors are errors in which two related words are not in the same number. Covert errors can be introduced when the related words are not required to have the same number (e.g. when using the determinant 'the').

Misselection (Agreement: Person) errors are errors in which two related words are not in the same person(e.g. 'I' and 'my').

Misselection (Agreement: Blend) errors are a combination of the three above.

Misselection (Contraction) errors are errors relating to obligatory use of contractions. There is no example of this in English. As an example, in EP you can not use the two words 'Em' and 'a' in a sequence without contracting them into 'Na'.

Misordering errors occur when the sentence has every word needed to convey the intended meaning, but the words' ordering was done incorrectly.

Semantic Level

Semantic level errors include errors regarding the source word's meaning on a target language. The error types included in this level and examples for each are in Table 2.4.

Error Type	Lang	Intended Sentence	Visible
Confusion of senses	EN	The box was full.	The cashier was full.
	EP	Ele poisou os óculos na mesa.	Ele poisou os copos na mesa.
Wrong Choice	EN	On New Year's eve I'm going to wear my best suit.	On New Year's eve I'm going to wear my best truck.
	EP	Na véspera de ano novo vou usar o meu melhor chapéu.	Na véspera de ano novo vou usar o meu melhor camião.
Collocation	EN	I want to catch the bus and take the pill.	I want to catch the bus and confiscate the pill.
	EP	Quero apanhar o autocarro e tomar a pílula.	Quero capturar o autocarro e tomar a pílula.
Idioms	EN	It's raining cats and dogs today!	It's raining pots today!
	EP	Está a chover a potes hoje!	Estão a chover cães e gatos hoje!

Table 2.4: Semantic level error examples

Confusion of Senses errors occur when a source language word has homonyms and is translated to the wrong meaning in a target language. In the example, the EP word 'caixa' was mistranslated into one of its homonym meanings, the cashier, instead of 'box'.

Wrong Choice errors occur when a source language word is translated to a non-related word in the target language.

Collocations as defined by James[3] according to Costa et al.[2] "[. . .] are the other words any particular word normally keeps company with". In other words, any sequence of words that together do not mean the composition of their constituting words(e.g. 'Get' and 'on', 'get' and 'up'). **Collocation*** errors occur when a collocation's constituting words are translated literally.

Idiom errors are the same as collocation errors but regarding language idioms.

Discourse Level

Discourse level errors do not necessarily break grammatical rules nor the meaning of the sentence but are not written in an expected way. The error types included in this level and examples for each are in Table 2.5.

Error Type	Lang	Intended Sentence	Visible
Style	EN	I need permission to be authorized to improvise.	I need authorization to be authorized to improvise.
	EP	Preciso de autorização para permitir tal loucura.	Preciso de permissão para permitir tal loucura.
Variety	EN	I'm seeing the most beautiful colors.	I'm seeing the most beautiful colours.
	EP	No seu discurso, João. . .	Em seu discurso, João. . .
Should not be translated	EN	Have you ever read a book written by Fernando Pessoa?	Have you ever read a book written by Ferdinand Person?.
	EP	Já alguma vez provaste uísque Johnny Walker?	Já alguma vez provaste uísque João Andante?

Table 2.5: Discourse level error examples

Style errors are errors that occur when the intended is conveyed but there was a bad stylistic choice of words (e.g. word repetition).

Variety errors are errors that occur when the intended meaning is conveyed and the sentence is grammatically correct in a variety of the target language (e.g. Brazilian Portuguese (BP) and EP).

Should not be translated errors are errors that occur when a word has been translated but it should not have been (e.g. movie title).

2.1.3 Unbabel Taxonomy

What Unbabel currently uses as their taxonomy mostly overlaps with the taxonomy described in 2.1.2. The main difference is that Unbabel also uses some error types relating to the translation job requirements, being more precise in some error types.

Accuracy errors

Accuracy errors are errors relating to word misplacement, misuse and absence. Mistranslation errors include errors regarding wrongly translated words and their cause. Mistranslation errors relate to the L2F taxonomy according to Table 2.6.

Unbabel Taxonomy		L2F Taxonomy
Mistranslation	Overly Literal	Collocation, Idioms and Confusion of Senses
	False Friend	Wrong Choice
	Should not have been translated	Should not be translated
	Lexical selection	Style

Table 2.6: Mistranslation errors

Overly Literal errors occur when a word in a source language is translated literally into another in a target language, like when included in a collocation, idiom or just a wrong translation of a word that has a homonym.

False friend errors occur when a word in a source language is similar to a word in a target language, resulting in a mistranslation. E.g English: ’eventually’, and EP: ’eventualmente’. ’Eventually’ means that something will happen, while ’eventualmente’ means that something might happen. A ’False friend’ error can be classified as ’Wrong choice’ error in the L2Ftaxonomy even if the ’Wrong choice’ is more generic than ’False friend’.

Should not have been translated errors occur when a word in source language was translated when it was not meant to.

Lexical selection errors occur when a word or expression is translated, albeit in an unusual way. E.g: Translating the English term ’Artificial Intelligence’ into ’Esperteza Artificial’ in EP (which roughly back translates to ’Artificial Smartness’).

Mistranslation errors are not the only errors that are included in Accuracy errors. Other accuracy error types and their corresponding error types in the L2F taxonomy can be found in Table 2.7.

Unbabel Taxonomy Type	L2F Taxonomy
Omission	Omission (function word) and Omission (content word)
Untranslated	Untranslated
Addition	Addition (function word) and Addition (content word)

Table 2.7: Other accuracy errors

Omission errors occur when a word is missing from a translation.

Untranslated errors occur when a word should have been translated but it was not.

Addition errors occur when a word should not be there but is.

Fluency errors

Fluency errors are errors that are not directly related to the text in the source language, but to the overall target language text correctness.

Unbabel Taxonomy Type		L2F Taxonomy
Inconsistency	Word selection	-
	Tense selection	Misselection (verb level:tense)
Coherence		-
Duplication		Addition (function word) and Addition (content word)
Spelling	Orthography	Spelling
	Capitalization	Capitalization
	Diacritics	Spelling
Typography	Punctuation	Punctuation
	Unpaired quote marks and brackets	Punctuation
	Whitespace	-
	Incosistency in character use	-

Table 2.8: Fluency errors

In Table 2.8 are the types of Fluency errors and their corresponding errors in the L2F taxonomy. **Inconsistency** errors occur when two identical words or expressions in the same text are not translated the same way. In **word selection** inconsistency, two equivalent expressions are not translated the same way. E.g: 'Mourning Tour' can be translated into both 'Passeio Matinal' and 'Volta Matinal' in EP. In **tense selection** inconsistency, two verbs meant to be in the same tense, are not.

Coherence errors are the most subjective of errors, occurring when the text is not clear to the interlocutor. Applies only to the text as a whole.

Duplication errors are a special case of the addition errors in section 2.1.3 which add a word or expression already present next to that same word or expression.

Spelling errors occur when a word is written incorrectly. **Orthography** spelling errors are those in which a letter or hyphen is added, removed, moved or substituted erroneously in a word. **Capitalization** spelling errors occur when a word has either a letter that should be a capital letter, but is not, or the inverse, a letter that should not be a capital letter, but is. **Diacritics** spelling errors are those in which a word has a missing diacritic, an extra diacritic or a combination of both.

Typography errors are those that affect the sentence structure. **Punctuation** typography errors occur when a sentence has misused punctuation, like a misplaced comma, a wrongly used question mark or even the lack of any punctuation. **Unpaired quote marks and brackets** typography errors occur when a sentence has either a quote mark lacking another quote mark or a bracket lacking an inverse bracket. **Whitespace** typography errors happen when a sentence has the either a lack of whitespaces or a surplus. **Inconsistent character use** typography errors are errors exclusive to languages that have a logographic writing system. The inconsistency lies in two equivalent words or expressions having a different character representing them.

Fluency **Grammar** errors are errors in which the used words have a semantic relationship to the correct words but are still wrong. Types of grammar errors and their corresponding L2F taxonomy error types can be found on Table 2.9.

Unbabel Taxonomy Type		L2F Taxonomy
Function Words	Prepositions	Addition (function word), Omission (function word), Misselection (agreement: gender), Misselection (agreement: number) and Misselection (agreement: blend)
	Determiners	Addition (function word), Omission (function word), Misselection (agreement: gender), Misselection (agreement: number) and Misselection (agreement: blend)
	Conjunctions	Addition (function word), Omission (function word), Misselection (agreement: gender), Misselection (agreement: number) and Misselection (agreement: blend)
Word Form	Part-of-Speech	Misselection (Word Class)
	Agreement	Misselection (verb level: person), Misselection (agreement: gender), Misselection (agreement: number), Misselection (agreement: person) and Misselection (agreement: blend)
	Tense/Mood/Aspect	Misselection (verb level: tense)
Word Order		Misordering
Sentence Structure		Style

Table 2.9: Fluency grammar errors

Function word errors occur when a preposition, determiner or conjunction is missing, misplaced, misused or in excess.

Word form errors occur happen when a word is translated correctly in semantic terms but nonetheless in a wrong way. **Part-of-speech** word form errors are those in which the word is in the wrong class, like an adjective instead of a noun. **Agreement** word form errors are errors in which two related words should share, but do not, the same number, gender, person or a combination of the previous. **Tense/Mood/Aspect** word form errors are errors that to not make the sentence grammatically incorrect but contradict the rest of the text in of those three terms.

Word Order errors are those in which the words have been misplaced within the same sentence.

Sentence Structure errors are similar to word order errors but do not make the sentence grammatically incorrect.

Style errors

Style errors are errors involving text formality and overall quality. Style error types and their corresponding types in the L2F taxonomy can be found in Table 2.10.

Unbabel Taxonomy Type	L2F Taxonomy
Register	-
Incosistent Register	-
Repetitive Style	Style
Awkward Style	Style

Table 2.10: Style errors

Register style errors occur when a text needs to be formal but is informal or vice-versa. **Inconsistent register** style errors happen when the text contains both informal and formal writing.

Repetitive style errors are those in which the text does not have varied vocabulary.

Awkward style errors are a last resource classification given to errors that do not fit any other possible errors but still make the sentence unfit for the desired end result.

Terminology errors

Terminology errors include errors related to vocabulary and/or style used in the text that do not meet the requirements provided by the entity that requested the translation job. None of the errors in this category have have a correspondence with any error type in the L2F taxonomy, being 'Style' the closest error type. There are only two errors in this category:

- **Noncompliance with client or company style guide**
- **Noncompliance with the glossary and vocabulary**

Other error types

Here we cover errors that did not fit in the previous categories. **Named entities** errors are comprised of errors regarding writing conventions of various entities. The errors covered are:

Person errors when referring to people.

Organization errors when referring to organizations (e.g. political, educational, governmental, etc.).

Location errors when referring to places.

Function errors when referring to professions.

Product errors when referring to products (e.g. car model, toy line).

Amount errors when referring to measurements (e.g. kilograms, pounds, centimetres, miles, etc.).

Time errors when referring to dates and time.

Wrong Language Variety errors are errors related to the usage of the wrong language variety in the translation, like using BP instead of EP or vice-versa. These fit into the 'Variety' errors in the L2F taxonomy explained in 2.1.2.

Lastly, **Formatting and Encoding** errors cover errors regarding text encoding(for incompatible characters) and formatting (e.g. tabs instead of spaces).

2.1.4 MQM Taxonomy

MQM[4] is short for "Multidimensional Quality Metrics", a framework for describing translation quality metrics. *MQM* is able to describe, coherently and consistently, translation quality metrics while tying them to specific project requirements.

While *MQM* is not limited to identifying translation quality issues, as it also measures design quality and formatting, it has a good coverage of diverse translation errors. In this section we will describe the parallelism between the previously discussed taxonomies and peculiar error types specific to the *MQM* taxonomy.

Similarly to the L2F taxonomy and the Unbabel taxonomy, the *MQM* taxonomy is sub-divided. The sub-divisions are called dimensions. Some of these dimensions, the Design dimension, the Internationalization dimension, Locale convention and Verity dimension are not related to linguistic errors. The Design dimension concerns the presentation of text, including issues with fonts, graphics, alignment and overall layout of the text. The Internationalization dimension include multiple issues concerning a product's internationalization, including inadequate forms for a country's language, inadequate security for encodings or even mentions to products unavailable in a given country. The Locale convention dimension includes issues concerning locale formatting, like phone number formatting, currency formatting and date formatting. The Verity dimension regards issues relating to inappropriate content for the target locale/audience, like unavailable options for a product in the given locale or the text includes unnecessary legal information for the target locale. These dimensions will not be further discussed in this document as they are not within the scope of this work.

Accuracy

Some of the errors types in the Accuracy dimension can be categorized in the Lexis Level of the L2F taxonomy, as can be seen in Table 2.11.

MQM Taxonomy	L2F Taxonomy
Addition	Addition(Function), Addition (Content)
Omission	Omission(Function), Omission(Content)
Untranslated	Untranslated

Table 2.11: Accuracy errors in the MQM taxononmy

An **Addition** error consists of the addition to the text of a word that was not in the original text.

Omission errors occur when content present in the original text is not present in the translation. Omission errors can be further specified into **Omitted Variable** errors, which are non-linguistic in nature as it concerns omitted placeholder variables. **Untranslated** errors consist of the presence of non translated content that should be translated in the translated text. **Untranslated** errors can be further specified as **Untranslated graphic** when parts of a graphic were not translated when they should have been.

Mistranslation errors are also present in the Accuracy dimension. They occur when wrongly translated content is present. **Mistranslation** errors can be further specified into:

- **Ambiguous Translation** consisting of ambiguous translations the translated content.
- **Date/Time**, consisting of dates or time not matching between the original and translated text.
- **Entity**, consisting of entities (e.g a city or name) that do not match between the original and translated text.
- **False Friend**, consisting of mistranslating a word into a similar, yet wrong, in the target language. These are identical to False Friend errors present in the Unbabel taxonomy.
- **Technical Relationship**, which is related to inaccurately translated relationships respecting technical knowledge.
- **Number**, consisting of inconsistencies in numbers between the original text and its translation.
- **Overly Literal**, which can also be found in the Unbabel taxonomy, consist of translations that are overly literal.
- **Should have not been translated**, which is the presence of translated content that should have not been translated in the translated text. Also present in the Unbabel taxonomy and L2F taxonomy (as `Should not be translated` error in the Discourse level).
- **Unit Conversion**, consist of wrongly converted units of measure between the original and translated text (e.g miles and kilometers).

Most of these Errors can be found in the Unbabel taxonomy, albeit in different categories, as seen in Table 2.12.

MQM Taxonomy		**Unbabel Taxonomy**
Mistranslation	Date/Time	Time errors
	Entity	Person, Organization, Location, Product
	False Friend	False Friend
	Number	Amount
	Overly literal	Overly literal
	Unit Conversion	Amount Errors

Table 2.12: Accuracy errors in the MQM taxonomy and in Unbabel's taxonomy

Also in the accuracy dimension are the **Over Translation** and the **Under Translation** in which a translation is either more specific than the original text or less specific, respectively.

Also in the Accuracy dimension are the **Improper exact TM match** issues, which are non-linguistic in nature, as they relate to wrong translations provided by a specific translation technology (Translation Memory system) is used.

Fluency

The Fluency dimension includes issues related to the form or content of a text. In Table 2.13 are the relevant errors in the Fluency dimension and their parallel errors in either the Unbabel taxonomy or L2F taxonomy, as is most appropriate for each case.

MQM Taxonomy Error Type	Taxonomy	Error Type
Ambiguity	-	
Ambiguity (Unclear Reference)	-	-
Character Encoding	Unbabel	Formatting and Encoding
Coherence	Unbabel	Coherence
Cohesion	Unbabel	Function Words
Duplication	Unbabel	Duplication
Grammar	Unbabel	Function Words, Misselection, Tense/Mood/Aspect
Grammar (Function words)	Unbabel	Function Words
Grammar (Word Forms)	L2F	Misselection
•Word Forms (Agreement)	L2F	Misselection(Agreement), Misselection (Verbs)
•Word Forms (Part of Speech)	L2F	Misselection(Word Class)
•Word Forms (Tense/Mood/Aspect)	Unbabel	Tense/Mood/Aspect
Grammar (Word Order)	L2F	Misordering
Grammatical Register	Unbabel	Register
Inconsistency	Unbabel	Inconsistency
Non-allowed characters	Unbabel	Formatting and Encoding
Offensive	Unbabel	Awkward Style
Sorting	L2F	Misordering
Spelling	L2F	Spelling
Spelling (Capitalization)	L2F	Capitalization
Spelling	Unbabel	Spelling (diacritics)
Typography	L2F	Punctuation
Typography (Punctuation)	L2F	Punctuation
Typography (Unpaired quote marks or brackets)	Unbabel	Unpaired quote marks and brackets
Typography (Whitespace)	Unbabel	Whitespace
Unintelligible	-	

Table 2.13: Fluency errors in the MQM taxonomy

Ambiguity errors are similar to **Ambiguous Translation** errors but differ on their cause, as **Ambiguous** errors are monolingual in nature. **Ambiguity (Unclear Reference)** issues occur when the text referential mechanisms (like pronouns) with an unclear reference. **Character Encoding** issues are related to garbled characters caused by the incorrect application of an encoding. **Coherence** issues are not dependent on a single sentence or word, but occur when the text as a whole is either inconsistent or does not make sense. **Cohesion** errors are the omission/non-correctness of text portions needed to connect the text as an understandable whole. **Duplication** errors occur when words, expressions or whole paragraphs are repeated unintentionally. **Grammar** issues are related to grammar or syntax and do not include punctuation or spelling errors. **Grammar (Function Words)** errors occur when function words are misplaced, omitted or misused. **Grammar (Word Forms)** errors are present when a word is malformed even when semantically related to the correct word. **Word Form (Agreement)** issues are present when a multitude words that are required to agree, (in either case, number, person or grammatical features), do not. **Word Form (Part of Speech)** errors occur when a word is wrongly classified as to their Part of Speech (POS). **Word Form (Tense/Mood/Aspect)** errors happen when a verb form is in the wrong mood, tense or aspect. **Grammar (Word Order)** issues occur when words are in the in the incorrect order. **Grammatical Register** errors happen when informal wording is used instead of formal wording and vice-versa. **Inconsistency** issues are present when the same entity is referenced as two different words or expressions in the text. **Non-allowed characters** issues are present when non allowed characters are in use. **Offensive** errors are related to the presence of offensive words, expressions or symbols in parts where there are meant to be none. **Sorting** issues occur when lists are not in the sequence they should be. **Spelling** errors are present when words are not in their correct spelling. **Capitalization** errors are those in which words begin with capitalized letters when they should not and vice-versa. **Diacritics** errors occur when words have either omitted, extra or misplaced diacritics. **Typography** issues are present when the mechanical representation of text if incorrect, not including spelling. **Typography (Punctuation)** errors are those where punctuation marks are used incorrectly. **Typography (Unpaired quote marks or brackets)** issues are self explanatory. **Typography (Whitespace)** errors are related to misused whitespace characters. **Unintelligible** errors are present when the error type cannot be determined as the text suffers from an major lack of fluency.

Corpus Conformance, **Index/TOC**, **Link/cross-reference**, **Pattern problem** are non-linguistic issues. **Corpus Conformance** occur when the text does not conform to a reference corpus. **Link/cross-reference** occur when links or or other non-linguistic references point to an erroneous or non-existent location. **Index/TOC** occur when there are errors in the index or table of contents. **Pattern problem** issues are present text conforms to a regular expressions that is not allowed.

Style

In the Style dimension the errors, as the implies, are related not to real errors that change the sentences' meaning but errors related to stylistic choices. In Table 2.14, we can find each Style error and their parallel in either the Unbabel taxonomy or the L2F taxonomy, as is most appropriate.

Register errors occur when the text is in formal language instead of informal and vice versa. **Register (Variants/slang)** errors occur when inappropriate slang for the register is used. **Awkward** issues exist when the wording is awkward. **Company style** errors are present when the style goes against company/organization guidelines. **Inconsistent style** errors exist when the style is inconsistent in a text. **Third-party style** issues exist when the text violates a third-party style guide. **Unidiomatic** errors are present when translations are grammatical but not idiomatic.

MQM Taxonomy Error Type	Taxonomy	Error Type
Register	Unbabel	Register
Register (Variants/slang)	Unbabel	Register
Awkward	Unbabel	Awkward style
Company style	Unbabel	Noncomplicance with client or company style guide
Inconsistent style	Unbabel	Awkward style
Third-party style	-	
Unidiomatic	Unbabel	Overly Literal, Lexical selection

Table 2.14: Fluency errors in the MQM taxonomy

Terminology

The Terminology dimension is related to the use of domain or organization specific terminology.

MQM Taxonomy Error Type	Taxonomy	Error Type
Inconsistent with termbase	Unbabel	Noncompliance with the glossary and vocabulary
Inconsistent with termbase (Company terminology)	Unbabel	Noncompliance with glossary and vocabulary
Inconsistent with termbase (Third-party termbase)	Unbabel	Noncompliance with glossary and vocabulary
Inconsistent with domain	Unbabel	Noncompliance with the glossary and vocabulary
Inconsistent use of terminology	Unbabel	Inconsistency (Word selection)
Inconsistent use of terminology (Multiple terms for concept in source)	Unbabel	Inconsistency (Word Selection)
Inconsistent use of terminology (Multiple translations for the same term)	Unbabel	Inconsistency (Word selection)

Table 2.15: Terminology errors in the MQM taxonomy

Inconsistent with termbase issues are present when a term is not consistent with the termbase.

Inconsistent with termbase (Company terminology) errors are present when the text violates company/organization terminology guidelines.

Inconsistent with termbase (Third-party) issues occur when the text violates third-party terminology guidelines.

Inconsistent with domain errors exist when a term is used contrary to the general domain expectations.

Inconsistent use of terminology issues are present when the same term is referred to in different ways.

The child issues of the latter error, **Multiple terms for concept in source** and **Multiple translations for the same term**, regard the same issue but refer to concepts and translated terms respectively.

2.2 Error Generation Systems

In the past decade there have been some error generation tools like GenERRate[1], Missplel[5] and Antispell[6]. Unfortunately Antispell is only briefly referred to in Agirre et al.[6] and as such will not receive the same focus as the other two systems in the following sections.

2.2.1 GenERRate system

GenERRate[1] is a language neutral, taxonomy neutral error insertion tool. It classifies every error as either an **Insertion**, **Omission**, **Substitution** or **Move** (as in Section 2.1.1) and lets the user customize error insertion through a configuration file. It allows to insert errors depending on POS-tags which permits a more precise way of inserting errors since we can place errors within specific word classes.

In each of the four error insertion types they provide the option of generating the error at random (e.g. omitting a random word in a sentence), considering a specific POS tag (e.g. moving a word tagged as a noun to another position) or considering a specific POS tag with specific POS tagged neighbors to the left and/or right. Move type errors are exceptions to the latter option.

Omission errors, as previously explained, can be generated by deleting a random word (e.g. deleting 'bus' from 'Get on the [bus]."), deleting a word with a specific POS tag (e.g. omitting a verb from "[Get] on the bus"') or by deleting a word with a specific POS tag neighboring words with specific POS tags (e.g. deleting a noun that has an article on its left from "Get on the [bus]).

Insertion errors are generated similarly to omission errors, but the POS specified is the tag the new word is going to have. Also, the user has the liberty to choose between inserting a word from a word list or from the sentence itself (e.g. inserting a word from the sentence after an article "Bob's burguers are the burguers best!").

Substitution errors, similarly to insertion and omission errors, can be generated while specifying both a POS tag for the removed word and the new word. The new word can either be specified or randomly picked from a word list. The removed word can also be specified (e.g. replacing 'car' with 'truck' in "I drove my spouse to the hospital in my car." resulting in "I drove my spouse to the hospital in my truck."). There is a different substitution error insertion method that is unlike the others and consists in replacing a word at random with the same word but in the wrong form when given a pair of two related POS tags (e.g. Noun Singular & Noun Plural, Adjective & Adverb... etc.). GenERRate changes the words through a series of rules. As an example, to transform an adverb into adjective, they have a rule that removes the suffix 'ly' from the word and verifies if it ends in 'i'. If so, the 'i' is then transformed to a 'y' (e.g. 'deeply' $\rightarrow$ 'deep', 'wearily' $\rightarrow$ 'weari' $\rightarrow$ 'weary'). These rules have exceptions, like 'simply', these are predicted in GenERRate. Wrong form substitution is the only insertion method in GenERRate that is not language independent.

Move errors can be generated by moving a word to random position. This word is either a random word from the sentence or a word with a specified POS tag. In these errors, instead of letting the user specify the neighboring words' tags, GenERRate accepts a number and a direction (left/right) as input that will define how the word will be moved (e.g. moving a noun two positions to the right in "My daughter thinks I am the best dad in the world.” can result in “My [] thinks I daughter am the best dad in the world.”).

The configuration file supplied to GenERRate must have a probability associated to each error type that will determine how often that specific error is inserted. Error insertion methods proposed in GenERRate are still valid whilst inserting error types in the L2F taxonomy referred to in Section 2.1.2, namely addition, omission and misselection errors (through wrong word form replacement errors). Every other error type in Section 2.1.2 falls into substitution errors or move errors in GenERRate, which means that these errors can not be added into text without deliberate user manipulation.

GenERRate views errors that are invisible as something undesirable and therefore, prevent some invisible errors by excluding tense errors in the wrong form substitution errors. In ErrorIST, we want to insert invisible errors, but we do not wish to do so inadvertently, which means this is a concern in visible error generation.

GenERRate’s source code is publicly available.

2.2.2 Missplel system

Missplel[5] is an error generation tool that was created to aid the creation of error annotated corpora. Missplel has four main modules for error introduction. The *Damerau* module, which introduces Damerau type errors as described in Damerau[7]. These errors are Spelling errors (as explained in 2.1.2) resulting from keyboard mistypes (e.g. ‘board’ and ‘borad’). Missplel, unlike GenERRate, has a flag that limits these errors, used in the *Damerau* module, giving the user control whether they wish to potentially insert invisible errors in the text by accidentally creating other plausible words. The second module is the *Split Compound* Module. This module, as the name implies, creates errors related to compound words like transforming ‘blackboard’ into ‘black board’. The third module is the *Sound Error* module, that fetches sound errors from a file and uses them as regular expressions to introduce them in text (e.g. transforming ’bee’ into ’be’). The fourth, the *Syntax* module, uses regular expressions which may take into account tags and words. The *Syntax* module can also be used for introducing word order errors, agreement errors, verb tense errors, repetition errors or omission errors. Unfortunately, due to the original paper’s lack of detail in the modules’ explanation, and the lack of comments in the code we cannot tell much more of the system’s error insertion methods. Unlike GenERRate[1], all these errors are only introduced randomly.

2.3 Replicating natural errors for Machine Learning

Most of the systems that used artificial error generation, used them for creating corpora to use with machine learning in order to either detect or correct errors. Izumi et al[8] inserted article errors in order to enlarge the corpus they were using. Sjöbergh[9] inserted compound errors (like the Split Compound module in Missplel) and word order errors in unnannotated corpora to improve their machine learning model. Foster[10] inserted spelling errors, omission errors, addition errors, agreement errors and verb errors in an existing treebank to create a treebank of ungrammatical sentences to provide training and test data for parsers. Wagner[11] also inserted the previous error types on a grammatical corpus to create training data for a classifier capable of distinguishing a grammatically correct sentence from an incorrect one. Rozovskaya and Roth[12] inserted errors in Wikipedia text in order to train error correction systems. Imamura et al[13] inserted errors in error prone text in order to improve features on the training set for an error correction machine learning application. Felice[14] also inserted errors on Wikipedia text using various errors and error distributions in real error annotated corpora. In machine learning one of the desired characteristics of training corpora is the similarity between it and the type of text it is going to be used on, in this case wrongly written text. Therefore, one of the concerns is to make training corpora as similar as possible to real error prone text.

In the following sections we will detail some systems that introduced artificial errors for machine learning applications. In the first section, we detail systems that introduce errors randomly (unguided error insertion) and in the second section, we detail systems that introduce errors following a methodology (guided error insertion).

2.3.1 Unguided artificial error insertion

Izumi et al.[8], trained a maximum entropy model for error detection on an error annotated corpus consisting of 1,300 transcribed audio-recordings of English oral proficiency interview tests called the Standard Speaking Test (SST). The tests were taken by Japanese English learners and the transcriptions were tagged for forty five different types of errors. However, in their experiment, they reduced their scope to the detection of 8 different error types. When faced with less than ideal results, Izumi et al.[8] tried to enlarge the corpus with errors in order to retrain the maximum entropy model. Noticing that, in the corpus, most errors resulted from article confusion, they decided to focus on article errors of the three modalities explained in Section 2.1.1, that is, article omission, article addition and article replacement. Introducing article errors provided mixed results: precision in detecting general omission errors (not only articles omissions) went down, but overall precision and recall went up, specially for article errors. These results proved that artificial error insertion can improve results. Motivating research further, errors were introduced randomly and were restricted to article errors.

2.3.2 Guided artificial error insertion

Rozovskaya and Roth[12] created an error correction system for articles. Their objective was to investigate whether artificial error generation could benefit automatic error correction. In the authors' perspective, the improvement obtained by Izumi et al.[8] could be due to the corpus size boost and not due to artificial error insertion. Furthermore, the error insertion was done randomly, which is not desirable, since the resulting training data will not resemble the test data. As such, Rozovskaya and Roth, in order to test the true benefits of artificial error insertion, tried artificial error insertion whilst following an error distribution resembling naturally occurring errors. One of the methods they used for error insertion, *General*, consists of error insertion at random with a probability of *(1 - x)*, where *x* is defined manually. Another method, *ArtDistrBeforeAnnot*, consists of replicating article distribution in error annotated corpora. In this method, Rozovskaya and Roth [12] also impose a manually defined minimum error rate for each different article. The third method, *ArtDistrAfterAnnot* is equivalent to the previous method, *Art-DistrBeforeAnnot*, but uses corrected error annotated corpora. The fourth method, *ErrorDistr* uses the distribution of errors for each article and attempts to replicate it. The annotated error corpora used for distribution analysis is composed from essays of English language learners from 3 different nationalities: Czech, Chinese and Russian. It is important to note that none of these languages has an article system, i.e a word class specifically used before a noun to indicate the type of reference the noun is making. Czech and Russian data is from the ICLE corpus[15] which is composed of essays written by advanced learners of English. Chinese data comes from the CLEC corpus which is composed of essays written by learners of English from all levels (unfortunately we could not find the corresponding publication for the CLEC corpus and therefore we could not cite it either). Errors were annotated and corrected in a portion of the data of each language by native speakers. Errors covered by this process are article errors, preposition errors, noun number errors, spelling errors, verb form errors and word form errors. When errors did not fit the previous categories, they fell into a category that includes word insertion/deletion and replacement errors (which, as explained in Section 2.1.1, accommodates every error type).

Rozovskaya and Roth extracted text from Wikipedia and inserted errors using the methods previously described, creating 4 different sets of training data, including the error free data. Using the Averaged Perceptron Algorithm[16] they trained 5 classifiers, one for each of the training sets. As testing set they used a different one for each language because the distributions were inserted differently in each of them. The best results coming from this approach are in Table 2.16. Results from article distribution methods (before and after annotation) were merged since the distributions were very similar.

Language	Classifier Training Data	Error Reduction
Chinese	ArtDistr	8.33%.
Russian	ErrorDistr	16.05%.
Czech	General	14.69%.

Table 2.16: Rozovskaya and Roth's[12] best results

The achieved results proved not only that training with artificial error corpora is useful but that taking into account error distribution is positively meaningful as well. With ERRORIST we aim to introduce realistic errors. Machine learning applications benefit from this realism as well since it makes the training data similar to data the application is going to be used on. It is our hypothesis that, if this approach benefited a machine learning application, it was because the errors inserted were similar to real errors. Taking this into account, we hope to create realistic errors by introducing errors in a similar way, by following error distributions of real error prone data during error insertion.

Rozovskaya et al[17] tried creating an article error correction system with a novel approach. Since most English speakers, native or not, use articles correctly 90% of the time, which means article errors are sparse. This sparsity, present in the training data for article error detection/correction systems, leads to a strong reliance on a feature always present in the article examples, the source word. In correct usages the source word always has the corresponding tag (e.g. 'A' is the label of 'the' in "Bob is a doctor."

and ‘A’ is also the label for ‘the’ in “I took the shower.”). In order to solve this overfitting problem, these authors propose the *error inflation method*. *Error inflation* reduces the proportion examples in which the source word and the label are identical (reducing the number of correct examples) and distributes the extra probability among typical occurring errors, boosting their numbers. According to Rozovskaya and his team: “This method causes the classifier to rely on the source feature less and increases the contribution of the features based on context.” Training the Averaged Perceptron algorithm[16] in an error inflated training set and subsequent testing lead to an overall increase in recall.

Felice and Yuan[14] tried to take controlled artificial error insertion even further, using distributions based on semantics, i.e PoS tags. Like Rozovskaya and Roth, they used Wikipedia articles for artificial error insertion. However, Felice and Yuan, unlike Rozovskaya and Roth, try to introduce open class errors. Open class errors are errors relating to open classes, like the noun or verb class[1].

Inserting open class errors in a controlled manner needs careful consideration that article error insertion did not. Nouns and verbs can often be used as one or another for example (e.g. verb ‘play’ and noun ‘play’), distributions used in error insertion must take the PoS information into account in order to prevent the errors meant for verb forms, from being inserted into nouns, representing inadequately naturally occurring errors.

Using the NUCLE[18] corpus, a corpus consisting of about 1,400 student essays from undergraduate university students at National University of Singapore (NUS) with a total of over one million words annotated with error tags and corrections, Felice and Yuan[14] obtained error distributions according to 5 interpretations.

- Error Distribution: probability of an error occurring within a relevant instance (the authors are not very clear as to what qualifies as a relevant instance besides a noun phrase).
- Morphological Distribution: distribution of words in a morphological context, like noun number or PoS tag (e.g. how many singular head nouns requiring the article ’an’ use other articles).
- PoS Disambiguation Distribution: distribution of words used in specific PoS tag disambiguation, like when dealing with the word ’play’ used as a verb or noun (useful for open class error insertion).
- Semantic classes Distribution: distribution of words used in a specific meaning context, like prepositions used while referring to a location or person.
- Word Sense Distribution: distribution of words used in a polysemous word context, like prepositions when referring to a bank as an institution or river bed.

Felice and Yuan tried creating a Statistical Machine Translation (SMT) system for error correction by training on 11 different sets of data: the aforementioned NUCLE corpus, a training set for each of the error insertion methods in Wikipedia, and every combination of the NUCLE[18] with one of the previous artificial error insertion sets. Error insertion used the *error inflation* method explained previously.

They trained a PoS-factored phrase-based model[19] on the different 11 training sets and tested them on the testing set of the NUCLE[18] corpus. The results varied depending on the training sets.

[1]Open classes. `https://en.wikipedia.org/wiki/Part_of_speech` Accessed last in 2016-01-08

Systems that just used either *Error Distribution* or *PoS disambiguation* for training improved precision at the expense of recall. This recall drop contradicts Rozovskaya et al, which demonstrate a recall improvement when using error inflation. Felice and Yuan state that this contradiction might be justified by differences in training paradigms and data. Systems that used hybrid datasets improved results overall, which suggests that artificial and natural occurring errors are not alternative, but complementary elements desired in corpora.

In ErrorIST, we want to provide plausibility to the errors generated through controlled error insertion. Desired error frequency is provided by the user.

2.4 Visible vs Invisible

Artificial error insertion that does not take into consideration the whole sentence, or at least some context, can create a **covert error**. Foster et al. [10] cited James [3] for his definition of covert errors. Unfortunately we could not acquire this book and verify it ourselves, but according to Foster et al., James defined covert errors as errors that result in a well-formed sentence under some interpretation different from the intended one. Covert errors can only be detected if some context is taken into account, like the remaining text or the translation source. For instance, if the word ‘three’ is misspelled as the word ‘tree’ in “I love my tree friends.”, the sentence remains grammatically correct, which means that, to detect this error, an editor must have prior knowledge that the friends the sentence is referring to, have no relation to trees. Examples for both these errors can be found on Table 2.17.

For the purpose of this work we will also define types of errors:

Visible Errors ; that is, artificial errors that are noticeable by making the sentence grammatically incorrect;

Invisible Errors ; that is, artificial errors that are intentional covert errors, i.e not noticeable unless a context is provided, because they only make the sentence incorrect semantically.

Intended Sentence	Visible Error Example	Invisible Example
Receiving presents is delightful!	<u>Recieving</u> presents is delightful!	Receiving presents is <u>awful</u>!
I like my new pants!	I like my new <u>pamts</u>!	I like <u>your</u> new pants!
I am a great person!	I <u>are</u> a great person!	I am a great <u>beautiful</u> person!

Table 2.17: Visible/Invisible error examples

GenERRate (described in Section 2.2.1) views errors that are invisible as something undesirable and therefore, prevent some invisible errors by excluding tense errors in the wrong form substitution errors.

One of the initial attempts of this Thesis was to automatically create Invisible errors but, since then, its aim has shifted to an accurate error insertion and correction retrieval tool. Examples for this concept in the L2F taxonomy can be found in Annex A.

2.5 Discussion

Artificial error insertion has been done before, but mostly for creating error prone corpora to be used in machine learning applications. None were used for linguistic evaluation, even if the generated errors aimed to be realistic. Currently available error insertion tools are generic and are limited regarding the error types they insert currently. With the exception of the English verb modifications made in *GenERRate*, the available error types fit best with the simple taxonomy described in Section 2.1.1. While there are many error taxonomies, each of the error types can be matched with the error types in the L2F taxonomy (with few exceptions) which is a taxonomy oriented towards machine translation errors. Errors that editors in Unbabel deal with, since they have to edit out the errors made by Unbabel's machine translation system, mostly reside in the error types described in the L2F taxonomy.

ERRORISTwill then both aim to create a wider variety of errors, beginning with those in Section 2.1.2, and to retrieve them for correction detection. While ERRORISTis meant to be a language neutral error generation tool, since no other system has done it, the currently implemented error types are refined for the EP language.

Chapter 3

Implementation

ERRORIST is composed of three modules: the `Error Generator`, the `Tracer`, and the `Evaluator` (Figure 3.1).

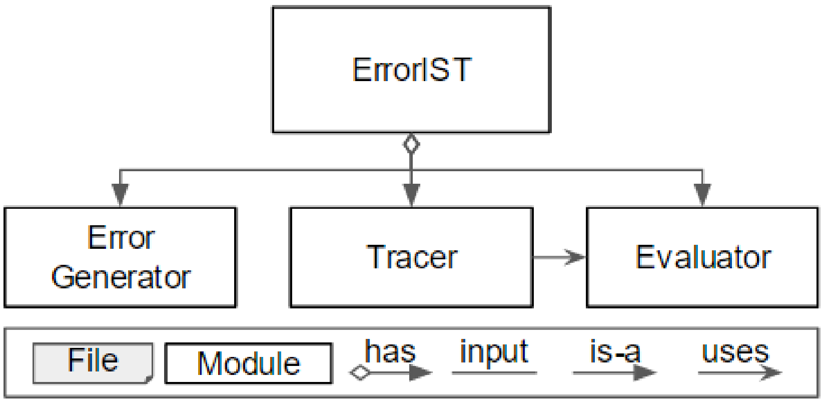

Figure 3.1: ERRORISTGeneral Architecture

`Error Generator` generates the errors, `Tracer` analyse how the generated errors were corrected and Evaluator grades the modifications detected by the Tracer. In the following sections we describe these modules. A detailed presentation of the generated errors is presented in Section 3.5.

Each and every module is extensible. Error insertion is not restricted to the methods already implemented in ERRORIST, which means the `Error Generator` should be extensible in order to accommodate new error types and the `Tracer` should be able to accommodate the new error types' relevant information and verification processes. A similar process can be applied to the `Evaluator`, as different purposes for ERRORIST's use may require distinct classification methods.

3.1 Error Generator

The `Error Generator` is able the generate errors from most of the L2F taxonomy's error types. Each error type has its own specialized Error Generator module in order to insert errors of that type, as can be observed in Figure 3.2.

Also seen in Figure 3.2, the `Error Generator` receives multiple arguments as input:

- a Sentence File, with a sentence per line, where errors will be inserted;
- a POS tagger for the appropriate language;
- a Word File, with words that will be used by `Addition Errors`;
- a Sound Confusion File, containing potential sound misinterpretations to be inserted by `Addition Errors`;
- a set of Affix Files with the suffix changes needed for `Misselection Errors`.

`Error Generator` can also receive additional arguments:

- Save file path, where the `Error data` will be stored, if none is provided, it will be stored in the program's execution folder by default;
- Error file path, where the file with errors will be stored, if none is provided, it will be stored in the program's execution folder by default.

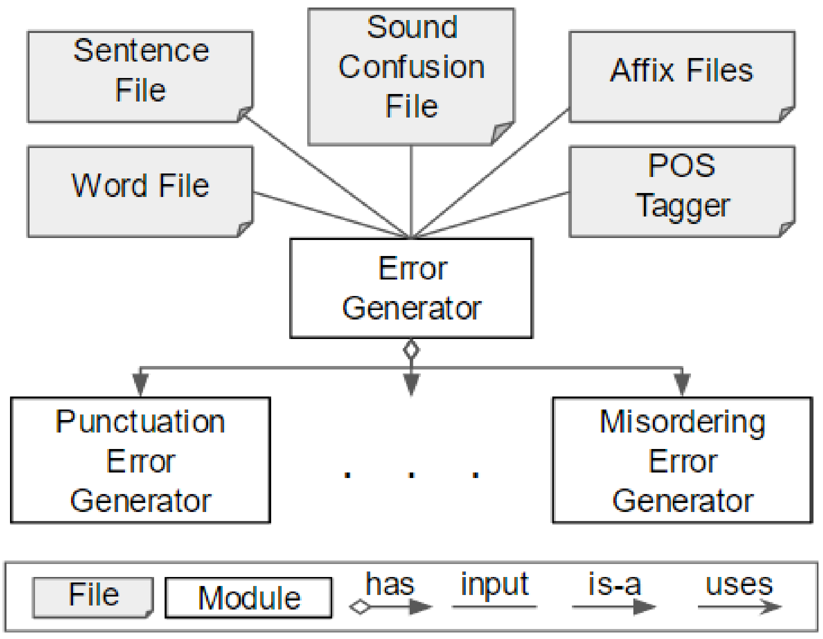

Figure 3.2: Error Generator Architecture

The `Error Generator` inserts an error per sentence and to do so, it distributes sentences to its specialized error generation objects in order to insert errors in them, also providing the necessary resources for that specific error's insertion. The specialized `Error Generators` then proceed to insert their specific error, producing two different outputs:

- The incorrect sentence
- A `Verification object`

If a certain error, for some reason, cannot be applied to the current sentence, it moves to the next sentence and that event is reported.

After producing the desired errors, the `Error Generator` stores all of the incorrect sentences and `Verification objects` as the `Error Data`, a structure to be used later by `Tracer`. While `Verification objects` are created by the `Error Generator`, they are further explained in Section 3.2, due to their importance in the `Tracer`'s functioning.

Besides this, two files are also created. The first file's purpose is to be corrected by the editor. It contains every sentence modified by the `Error Generator`. The sentences are numbered as to identify which `Verification object` is linked to them.

The second file's purpose is to be used as a reference. It contains the same content as the previous file with the addition of each error's type and the original sentence.

3.2 Tracer

The `Tracer` receives as input the `Correction File` (a file containing all the sentences modified by the editor) and uses the `Error Data` created by the `Error Generator` in order to identify changes in the sentence made by the editor. When an error is created, the relevant information is stored within a `Verification object`. What information is considered relevant varies from error type to error type, although most error types store the error placement, the original sequence, and the altered sequence. As an example, `Spelling Verification objects` contain the original word, the misspelled word and the word's placement in the sentence. The misspelling in "Bobby cried **wlof**!" would result in a `Verification object` storing "wolf", "wlof" and '2' as it is its position in the sentence (counting from 0). Due to their varying nature, each error type also has a specialized `Verification module`, much like the `Error Generator`, as can be observed in Figure 3.3.

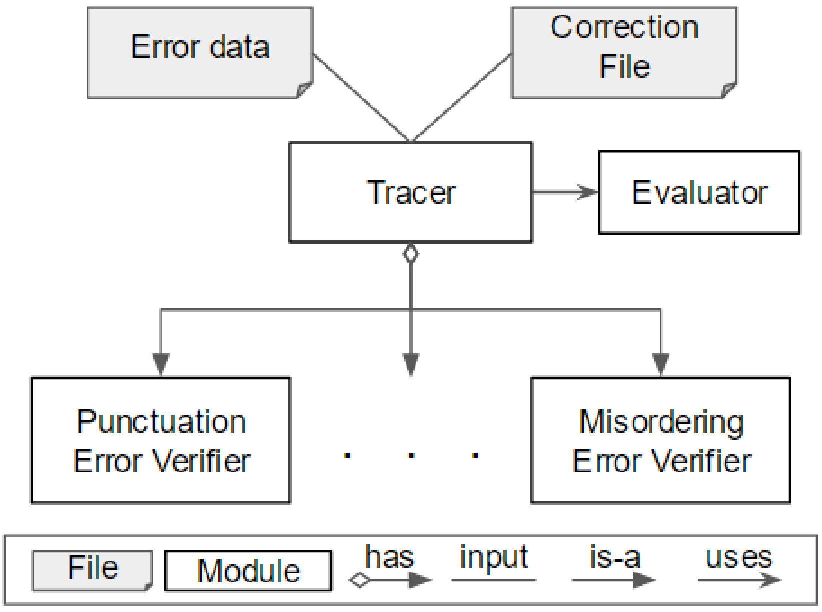

Figure 3.3: Tracer Architecture

The `Tracer` uses the `Verification objects` stored in the `Error Data` in order to identify changes in the sentence. As of now, the `Verification objects` identify the changes by checking the modified sentence for three aspects:

Changed? Did the editor modify the inserted error?

Expected? Did the editor modify the inserted error to its correct, original form?

Other? Did the editor modify the sentence in a way that does not directly affect the error?

As an example, consider the sentence ”Mary chased a lamb.”, which was modified by ERRORIST into the following sentence, ”ary chased a lamb.” through a `Spelling error` in the word ’Mary’.

If the editor modifies it back to ”Mary chased a lamb.”, the error has been **Changed** and the modification was **Expected**. **No Other** modification has been made.

If the editor modifies it further to ”Mary chased a little lamb.”, the error has been **Changed** and the modification was **Expected**. **Other** modifications have been made, as the word ’little’ was added.

If the editor modifies it to ”Gary chased a lamb.”, the error has been **Changed** and the modification was **Not Expected**. **No Other** modification has been made.

If the editor does not modify it, leaving the sentence as ”ary chased a lamb.”, the error was **Not Changed**. **No Other** modification has been made.

3.3 Evaluator

The `Evaluator`’s purpose is simple: to consider the modifications identified by the `Tracer` and to produce an adequate evaluation according to them. As such, its behavior can be summarized in the following input/output table (Table 3.1).

Changed?	Expected?	Other?	Result
Yes	Yes	No	Ok
Yes	Yes	Yes	Undef
Yes	No	No	Undef
Yes	No	Yes	Undef
No	–	No	Ko
No	–	Yes	Undef

Table 3.1: `Evaluator` behavior

An **Ok** evaluation means that the correction does **not** need no further human verification as it is unquestionably **Correct**. A **Ko** evaluation means that the correction does **not** need further human verification as it is unquestionably **Incorrect**. An **Undef** evaluation means that the correction **needs** further human verification as the `Evaluator` could not determine if the correction was correct or incorrect.

For example, if the sentence “Mary chased a lmb.” is corrected to “Mary chased a lamb.”, it is classified as **Ok** as it was both **Changed** and **Expected**, having no **Other** modifications made to it. If it was corrected to “Mary chased a lion.” It would be classified as **Undef** because, even though it was **Changed** and no **Other** modification was made to it, the change was not the **Expected** one. In another correction attempt, “Mary chased a little lamb.”, the sentence is classified as **Undef** as well, because even though the error was **Changed** and the change was **Expected**, there were **Other** modifications made.

3.4 Resources

While some errors can be introduced without the addition of any new content(e.g `Omission Errors`, `Misordering Errors`), others require either non-trivial modifications to words or new words altogether(e.g `Misselection: Verb Tense Errors`, `Misselection: Gender Agreement`, `Addition Errors`). For this reason, ERRORISTrequires some resources to properly introduce errors in the L2F taxonomy. In this section, we will discuss the motivation behind each of the resources used in ERRORIST.

3.4.1 POS tagger

One of the resources used is the POS tagger. The POS tagger is essential to most error types. Whilst no POS tagger is provided with ERRORIST, it uses one provided by the user. The POS tagger should be a variation of NLTK’s[20] POS tagger and the tag set used should be provided to ERRORIST. ERRORIST will later use the tagger in order to identify relevant word types during error insertion.

The necessary variety of the tags is the following:

Adverb, Noun, Adjective, Verb, Pronoun, Article/Determiner, Preposition and Conjunction.

Unfortunately, due to time limitations, this tag set is hard coded into ERRORIST. However, it was limited to one file, which means it can be easily refactored to accept it as user input.

3.4.2 Affix Files

Most `Misselection` errors are inserted by altering the word’s suffix according to rules provided by the user according to Ispell[21] entries. These entries are comprised of a regular expression, a removal sequence, a substitution sequence and information regarding the suffix change (like word number or verb tense).

```
regex > -removal, substitution ; "info_a=x, info_b=y"
```

ERRORIST also creates a reverse entry for each and every rule in order allow suffix changing in both directions. For example, if the file only contains entries regarding suffix changes when turning a verb from the infinitive to each possible tense, changing the verb’s tense would be impossible. Creating reverse entries allows ERRORISTto generate transition points that can be used to further alter the word. For example

```
I R > -R,AM # "P=3,N=p,T=pi"
```

It means that, to use this rule, the word must match “I R” at its end. Then, to alter it, the letter ’R’ must be removed and replaced with the letters “AM”. Using the rule to alter the word ”dormir” (to sleep), which is possible because it ends in “I R”, we remove the letter ’R’ from its end and replace it with “AM” which finally results in “dormiam” a past variant from the verb “dormir” (to sleep).

While these entries permit ERRORISTto alter verbs in tense and person, it does so using only the infinitive form, which is not ideal since most verb forms present in sentences are not in the infinitive form. For this reason, ERRORISTalso creates a reverse entry for each of these rules in order to make sure each verb form can return to its infinitive form.

As an example the reverse entry for the entry above would be:

```
A M > -AM, I # "P=3,N=p,T=pi,Reverse=yes"
```

Using the infinitive form as middle ground through these reverse entries, ERRORISTcan change a verb tense or person through its infinitive form. Using this reverse entry to transform “dormiam” into “dormir”, the verb’s infinitive form, and then using another entry like:

```
I R > -R,STE # "P=2,N=s,T=pp"
```

We successfully transformed “dormiam” into “dormiste”(second person singular form of the past perfect) using the verb’s infinitive form as a middle point.

As stated previously, most `Misselection` errors depend on affix files. These files have entries, like the ones above, that allow the modification of words through their suffixes. Despite this fact, not every modification is adequate for every word. It makes no sense to use an entry designed to transform adjectives into nouns if the word is already a noun for example. To cope with this situation, ERRORISTdoes not use a single affix file, but a set of affix files made out of the original, separated by purpose. ERRORISTuses a different file for: `Misselection: Verb`, `Misselection: Number`, `Misselection: Gender`, `Misselection: Word Class`.

This permits ERRORISTto apply the correct rules to the correct words. Each error that needs so, uses the POS tagger in order to distinguish which words should be considered candidates for error generation. For example, `Misselection: Verb Tense` chooses a random verb from the sentence and applies the entry as described above.

`Misselection: Word Class` works a bit differently, as its whole purpose is to be utilized on multiple word classes. By separating the affix files in categories and reading the categories’ headers ERRORISTcan use the appropriate rule to alter the a candidate word’s class (assuming it was correctly tagged). An example category looks like this:

```
flag *C:; "CAT=v,T=inf" #cao
T A R > -TAR,ÇÃO;
"CAT=nc,G=f,N=s,FSEM=cao"
CIONAR > -IONAR,ÇÃO;
"CAT=nc,G=f,N=s,FSEM=cao"
AIR > -IR,CÇÃO;
"CAT=nc,G=f,N=s,FSEM=cao"
UIR > -IR,ÇÃO;
"CAT=nc,G=f,N=s,FSEM=cao"
```

By interpreting the first line we conclude that this category’s purpose is to alter words from verb to another class. Unfortunately, the file is static which means the tag here should be in accordance with the tags used by the POS tagger.

Categories do not exempt `Misselection: Word Class` errors from filtering the words in order to prevent erroneous modifications (e.g changing an article's class), even if they are unlikely to occur.

3.4.3 Sound Confusion File

Spelling errors are not restricted to simple letter misplacements, omissions or repetitions. Homophone sequences are also a probable cause for erroneous spelling, caused by either similar sounding sequences (e.g confusing "ch" with "sh") or Qwerty keyboard mistypes (e.g typing an extra 't' when pressing 'y', due to its proximity). For this reason, ERRORIST accepts a `Sound Confusion File` for useful sequence modifications. If such a sequence is present in a sentence during the insertion of a `Spelling Error`, ERRORIST will use the file to replace the sequence with its homophone. The resource must be provided by the user and the format should be two homophone sequences per line, separated by any number of whitespace characters, as an example:

```
'ss'      'ç'
'c'       'k'
```

While this confusion file was made with homophones in mind, any two sequences the user considers adequate can be used. Such as 'hgt' and 'ght'.

3.4.4 Word File

For both `Addition Errors`, ERRORIST needs to introduce new words to the sentence. For this purpose, ERRORIST needs a `Word File`, a file composed of numerous words to introduce in a sentence. Each line of this file should have a tag (one of those used by the aforementioned POS tagger) as its first element and then all the words that can be classified as belonging to that tag. As an example:

```
ART a the an
NOUN dog cat car log
VERB bark wait comb
```

3.4.5 Contraction file

As mentioned in Section 3.5.3, `Misselection: Contraction errors` need a file composed of decomposed contractions in order to be inserted in a sentence. These files are be provided by the user and should have a contraction followed by its composing parts per line, as such:

```
contraction part1 part2
```

As an example, the EP contraction word "dessas" is formed by a contraction of "de" and "essas". In the file it should be written as:

```
dessas de essas
```

3.4.6 Person Agreement File

Since word candidates for `Misselection: Agreement Person` errors are not very diverse, its insertion relies on a small file containing potential changes to these words. Each line should contain all the candidate replacements in each number. As an example, take these two lines from the EP file:

```
meus teus seus nossos vossos
meu teu seu nosso vosso
```

3.4.7 Wiktionary

Wiktionary[22] is a collaborative project which aims to create a free multilingual dictionary. The purpose is to have all words of all languages described in a given language. There are currently ten different "Wiktionaries": English, Russian, Polish, Spanish, Italian, French, German, Greek, Japanese and Portuguese. As an example, the English *Wiktionary* should have a page not only for "dog" but for "cão" (dog in EP) and "chien" (dog in French), even if they are not words in the English language. These foreign words are described in English, which permit the user to associate both "cão" and "chien" to "dog".

Wiktionary is more of a dictionary than a translator, as its name implies. Nonetheless, its pages provide translations in numerous languages for each native word available. These translations, foreign words, also have pages of their own with their meaning written in the *Wiktionary*'s native language. For the purpose of ERRORIST, it will be used as a translator in order to produce `Confusion of Senses` errors. As an example, using the English *Wiktionary*, the English word 'chocolate' is translated to the French word 'chocolat' which, in turn, can be translated back to English as either 'chocolate', 'deceived' or tricked. By replacing the original English word, 'chocolate', by either 'deceived' or 'tricked', we can successfully create a `Confusion of Senses` error.

3.5 Error Insertion

In this section is detailed each error type's insertion method, limitations and resources used.

3.5.1 Orthography level

Considering `orthographic errors`, `capitalization errors` are inserted by changing a word's initial letter (from capitalized to non-capitalized and vice-versa), and `punctuation errors` are created by simply replacing a punctuation mark with another punctuation mark. The latter method proved to be inadequate according to Unbabel, as, according to them, they did not represent actual punctuation errors. For this reason, `punctuation errors` were further refined into other generation methods. One method generates `Punctuation` errors by adding a comma or a semicolon between a subject and predicate or between a verb and its complements. To introduce the latter, ERRORIST uses the POS tagger, identifies nouns or pronouns (the subject) and then identifies a verb (the predicate) placing a comma before it.

The same logic is applied when inserting a comma between the verb and its complements, where the verb is identified and if any adjective, noun, pronoun or article is found after it, a comma is placed after the verb. ERRORIST also inserts punctuation errors by removing a comma after an adverb. This is made possible by the POS tagger as well. ERRORIST first identifies an adverb and then checks the succeeding character for a comma, if found, it removes it. Finally, `spelling errors` are inserted by omitting, replacing, adding, or misplacing letters from a word. Considering that some errors are more plausible than others, either due to the involved language, or the used keyboard, ERRORIST also allows the user to add rules that will trigger specific spelling errors. This is possible through the `Sound Confusion File`, described in Section 3.4.3.

3.5.2 Lexis level

`Omission errors` are created by removing a word from the sentence; `Addition errors` are created by adding a word to the sentence. Both of these errors can be further specified, regarding whether a function word (e.g a conjunction) or a content word (e.g a noun or verb) should be omitted/added. To identify a word's class ERRORISTuses a POS tagger (described in Section 3.4.1). To add a word, ERRORISTuses the word's POS tag and searches the `Word File` (as described in Section 3.4.4) provided by the user for an adequate addition.

As an example of an `Addition error` (function), take into account the sentence " I bought a house.". ERRORIST will first search the `Word File` for a line with an appropriate tag, "ART", and, then, choose a random word from that same line. Choosing the word "the", ERRORIST would then add it to a random position like so: "I the bought a house.".

3.5.3 Grammar level

Misselection: verbs

`Verb Misselection errors` use the `Affix File` described in Section 3.4.2. During their insertion, ERRORISTuses the information provided in each entry (tense and/or person) to choose an appropriate entry to change the suffix.

Verb Tense errors make sure to change suffix using entries that maintain its person. Verb Person errors change suffix using entries that maintain its tense and, finally, Verb Blend errors change suffix using entries that keep neither tense, nor person.

An unfortunate limitation of using affix files to alter verbs, is its incapability to modify irregular verbs correctly. While there are entries regarding irregular verbs, ERRORISTHas no way to distinguish them from regular verbs.

As an example, the verb "ir" (to go), is an irregular verb but the following entry could still be applied:

```
I R > -R,STE # "P=2,N=s,T=pp"
```

This would result in the word "iste" which is not a word in EP

Misselection: word class

Word Class errors depend on the categories provided on the affix file, as described in Section 3.4.2. Each category corresponds to a possible word tag. A word with a matching tag will be changed to another category using an entry from that category.

Some examples of a word class entry are:

```
flag *n:  ; "CAT=v,T=inf"
I R > -IR,ENTE ; "CAT=adj,N=s,FSEM=nte"
O R > -R,NENTE ; "CAT=adj,N=s,FSEM=nte"
```

From the first line, this section comprises transformations applicable to verbs. Each entry has its own category with the word class from the resulting word as its value. As an example, using the first entry we can turn the verb “aderir” (to adhere) into the adjective “aderente” (adherent).

Misselection: agreement

Agreement Number errors are inserted by finding an appropriate entry to change the word’s number, as described in Section 3.4.2. Agreement Gender errors, due to the lack of entries related to gender switches in plural words, first turn the word to singular (if needed), then change the word’s gender, and finally back to plural form. Using as an example, the word “pato” (duck) and the following entry:

```
[^Ã][^LSMRNZX]> S # "N=p"
```

By adding the letter ‘s’, it would become “patos” (ducks). If there was the need for changing this word’s gender, ErrorIST would first turn into singular by using the above entry’s reverse entry. Then, using the following rule to change its gender:

```
[^Ã] O > -O,A # "G=f"
```

The word becomes “pata” (female duck). In order to maintain the word’s original number, ErrorIST uses the first entry again, turning “pata” into “patas” (female ducks).

Agreement Person errors, unlike the above Misselection errors, does not rely upon affix files but instead relies on a Person Agreement File since the potential word changes are very few. The file should be provided by the user and each line like the following example:

```
meus teus seus nossos vossos
meu teu seu nosso vosso
```

Where each line has every variation possible for a possessive determiner.

Misselection: contractions

Contraction errors are inserted by finding a contracted word and separating it using a file composed of contractions and their respective composing parts. The file’s formatting is described in Section 3.4.5.

Misordering

Misordering errors are inserted by misplacing a word, randomly chosen, within the sentence. Two positions are randomly chosen, and the word in the first position is switched to the second. If, for some reason, the sentence remains unchanged (repeated consecutive words for example), the positions are picked again until the sentence is altered.

3.5.4 Semantic level

While the Costa taxonomy is aimed towards translation errors, most of its error types do not necessarily have a bad translation as their cause. As an example, the misspelling of the word “dog”, “dawg”, does not have to be the result of a mistranslation of the EP word “cão”, it can happen by misspelling the word “dog”. Conversely, a Confusion of Senses error can not happen outside of a translation context. As an example, using the word “cashier” instead of the “box” can only be considered as a `Confusion of Senses` error if you also consider its original wording in EP, “caixa”. Until now, every error type discussed is inserted without taking into account a secondary language, but in Semantic level, this is not possible. With the exception of Wrong Choice errors, every other error in the Semantic level has a bad translation as its cause, which means we have to take into account the source language. To this end, we used Wiktionary[22]. Wiktionary is not a translation tool, but it serves the purpose of inserting these types of errors adequately. In `Confusion of Senses errors`, ERRORISTuses a language provided by the user in order to search for an adequate translation in the corresponding Wiktionary page. Then, it uses the translated word’s page to fetch all its meanings and returns one of the non-intended meanings. As an example, the English word ’chocolate’ is translated to the French word ’chocolat’ which, in turn, can be translated back to English as either ’chocolate’, ’deceived’ or tricked. By replacing the original English word, ’chocolate’, by either ’deceived’ or ’tricked’, we can successfully create a `Confusion of Senses` error.

Chapter 4

Evaluation

Error insertion is a simple task if quality is not a concern. As it is, ERRORIST must submit itself to evaluation, quantifying the quality of its error insertion and its ability to retrieve error corrections. Each correction classification follows the rules described in Section 3.3. In order to evaluate ERRORIST's quality, we must submit it to various tests. In the following sections we discuss the methods used for ERRORIST's testing, their reasoning, and, finally, their results. Every error in this chapter was generated as described in Section 3.5.

4.1 Error Quality

ERRORIST inserts errors through insertion methods aiming to replicate error types found in the L2F taxonomy described in Section 2.1.2. Whether ERRORIST is able to perform such insertions correctly or not, needs to be ascertained.

As an example, a Misselection: Verb Tense error inserted in the verb "ter" (to have) could be inserted successfully by transforming it into "tinha" (had), which can be inserted by ERRORIST. Also inserted by ERRORIST is the transformation into 'te', which, while a word in the EP language, is not a tense variation of "ter".

In order to evaluate ERRORIST's error insertion methods, we asked one of the L2F taxonomy article writers, Ângela Costa, to classify 6 ERRORIST generated errors of each error type regarding whether or not they were according to the taxonomy described in Section 2.1.2.

As observed in Table 4.1, most errors accurately represent the L2F taxonomy error types at least 66.67% of the time (at least 4 out of 6 errors). Half of the `Word Class` and `Verb Tense` errors accurately represent their error typing. And `Verb Blend`, `Agreement: Gender`, `Agreement: Blend` and `Confusion of Senses` errors fail to represent their error typing accurately more than 33.33% of the time (2 out of 6).

	According	Not According
Punctuation	6	0
Punct Add	6	0
Punct Omit	6	0
Capitalization	6	0
Spelling	6	0
Omission F	6	0
Omission C	5	1
Addition F	6	0
Addition C	6	0
Word Class	3	3
Verb Tense	3	3
Verb Person	4	2
Verb Blend	1	5
Gender	1	5
Number	4	2
Person	6	0
Blend	2	4
Contraction	6	0
Misordering	6	0
Confusion	1	5
Total	84	42

Table 4.1: Error Quality

4.2 Traceability

In order to ascertain whether the error corrections can be detected automatically we used two different texts for error insertion.

The first text is a story written in EP, and the second text is composed of translated film subtitles, each of the EP sentences was accompanied by the correct sentence in its native language, English. In order to provide varied errors for correction, we inserted 6 errors of each of the 21 types in the text, totaling 126 error prone sentences.

We extracted 126 sentences of each text to insert errors. Each of these sentences was automatically tagged using a bigram tagger from NLTK[20] trained on the Floresta Corpus[23]. Each file had 126 error-prone sentences, each of the sentences was accompanied with a number identification and, in the Subtitle file, the original error-free sentence in English. We then assembled 10 persons and divided them randomly into to two groups of 5, one for each file. Every one of them is a native EP speaker with and has a good understanding of the English language. They were provided with instructions to correct the sentences in order for them to make sense. A breakdown for each error type in each file can be found in Table 4.3 and Table 4.4 for the Story file and the Subtitle file respectively.

	Changed?	Expected?	Other?	Ok	Ko	T(%)
Story	454	411	163	334	110	70.48
Subtitle	489	386	210	325	73	63.17
Total	943	797	373	659	183	66.83

Table 4.2: Total scores

	Changed?	Expected?	Other?	Ok	Ko	T(%)
Punctuation	29	25	5	23	0	76.67
Punct Add	14	14	5	12	13	83.33
Punct Omit	14	14	8	13	9	73.33
Capitalization	20	20	6	19	5	80.00
Spelling	26	24	9	17	2	63.33
Omission F	13	8	10	6	14	66.67
Omission C	15	8	10	8	9	56.67
Addition F	30	30	2	28	0	93.33
Addition C	25	25	4	23	3	86.67
Word Class	25	25	5	22	3	83.33
Verb Tense	27	24	2	23	2	83.33
Verb Person	28	27	7	21	2	76.67
Verb Blend	21	14	10	12	2	46.47
Gender	20	18	4	16	9	83.33
Number	14	14	7	10	13	76.67
Person	16	15	10	13	7	66.67
Blend	21	21	12	13	5	60.00
Contraction	22	21	14	11	5	53.33
Misordering	30	20	12	12	0	40.00
Confusion	14	14	14	9	7	53.33
No Error	30	30	7	23	-	76.67

Table 4.3: Story file scores

	Changed?	Expected?	Other?	Ok	Ko	T(%)
Punctuation	26	16	8	15	4	63.33
Punct Add	23	21	6	19	5	80.00
Punct Omit	12	11	6	9	15	80.00
Capitalization	14	14	10	10	10	66.67
Spelling	27	22	9	20	0	66.67
Omission F	25	16	12	13	4	56.67
Omission C	25	19	11	19	0	63.33
Addition F	30	27	10	18	0	60.00
Addition C	30	22	13	17	0	56.67
Word Class	25	22	10	20	0	66.67
Verb Tense	27	19	9	18	1	63.33
Verb Person	19	5	12	4	6	33.33
Verb Blend	22	20	13	16	1	56.67
Gender	28	23	7	21	2	76.67
Number	19	17	15	14	1	50.00
Person	14	11	10	9	11	66.67
Blend	26	19	6	19	3	73.33
Contraction	19	16	13	12	4	53.33
Misordering	27	19	12	15	3	60.00
Confusion	21	17	12	13	3	53.33
No Error	30	30	6	24	-	80.00

Table 4.4: Subtitle file scores

4.2.1 Evaluation with Unbabel

In order to ascertain ERRORIST's capabilities as an evaluation tool we submitted it to an actual business environment test. To accomplish this, with Unbabel's support and cooperation, we assigned a fake editing task to 17 editors with some errors Unbabel's professionals deemed adequate in representing a good coverage of frequent errors their editors face while correcting translation tasks.

The text sample in which the errors were generated, kindly provided by Unbabel, was constituted by an e-mail letter translation with an error of each of 6 error types in ERRORIST:

Punctuation, Agreement: Verb Tense, Confusion of Senses, Capitalization, Contraction and Spelling.

The sample had 7 sentences, in which one was deliberately made to have no error. This sentence is included in order to ascertain how much editors edit a sentence even if it has no error.

While the errors were created by ERRORIST, 7 samples were created until one demonstrated adequate quality for Unbabel's needs. In Table 4.5, we can observe the quality of the corrections made by the editors.

	Changed?	Expected?	Other?	Ok	Ko	T(%)
Punctuation Addition	12	12	4	9	4	76.47
Capitalization	12	12	7	8	2	58.82
Spelling	10	10	8	9	0	52.94
Verb Tense	12	9	7	9	1	58.82
Contraction	13	13	8	9	0	52.94
Confusion	11	11	9	7	1	47.06
No Error	17	17	6	11	0	64.71
Total	87	84	49	62	8	58.82

Table 4.5: Unbabel scores

While the traceability is lower than the evaluation in Section 4.2, it still means more than half of the corrections (58.82%, as seen in Table 4.5) can be detected automatically.

4.2.2 Discussion

As we can observe in Table 4.2, only around 66.83% of corrections can be labelled as either correct or incorrect in a straightforward way.

We can interpret **Changes** as the number of times the inserted error was actually noticed, since it was modified (correctly or not). In the subtitle file the number of **Changes** corrections is slightly higher, probably due to the added context the original sentence provides. Conversely, the number of **Expected** corrections is lower for the same reason. Every translation presented was correct, still, the editors sometimes disagreed with the present translation and refined it to their terms, reducing the number of **Expected** corrections and raising the number of **Other** changes made in the sentences (as can be seen in Table 4.4).

The context, however, raised the number of both **Changed** and **Expected** significantly in `Omission` errors, as seen in Table 4.3 and 4.4, which indicates the error type's difficulty of detection outside of translation scenarios.

As we can observe in Table 4.2, the values on the tables for both files are very similar. The number of **Changed** and **Other** is slightly higher for the subtitles file. The traceability, the number of corrections ERRORIST can confidently determine as either correct or incorrect, for these errors (**T(%)**) is higher in the Story file, as well as the number of **Ko**. The number of **Expected** corrections for `Verb Person` errors in the Subtitle file is severely lower compared to the story file, again due to disagreements regarding the correct translation. They also differ in `Omission` errors, that have less *Corrected* instances, most likely due to the lack of context, the word was not guessed accurately.

In Unbabel's evaluation the results were similar to the Subtitle file's results, except for the `No Error` which proved to be over-edited by Unbabel's crowd editors.

Regarding error quality, as expected, the errors belonging to more complex error types have a lower reliability in their generation methods, as seen in Table 4.1. `Verb Tense`, `Verb Person` and `Verb Blend` all rely on `Affix files` for their generation methods. While `Affix files` are expressive regarding their suffix changes, they can also be used wrongly.

An unfortunate limitation of using this approach to alter verbs, is its incapability to modify irregular verbs correctly. While there are entries regarding irregular verbs, ERRORIST has no way to distinguish them from regular verbs.

As an example, the verb "ir" (to go), is an irregular verb but the following entry could still be applied:

```
I R > -R,STE # "P=2,N=s,T=pp"
```

This would result in the word "iste" which is not a word in EP.

Another probable cause is the ambiguity regarding which affix entry to use. If there is more than one entry fitting the suffix needed, ERRORIST chooses one them randomly, with the possibility of using a non ideal entry, resulting in less than adequate changes to regular verbs. `Agreement: Blend` errors also suffer from the same ambiguity regarding which affix entry to use.

`Confusion of Senses` errors rely on Wiktionary [22] to generate them. The current implementation has 2 major issues. One being the randomness with which ERRORIST chooses the candidates for this error type's insertion. While Content words normally have different meanings to them, this is not the case with Function words, and they can be chosen just as easily. As an example, the EP article "o" is translated to "the". Translating it back to EP results in either "o", "os", "a" or "as", which are (excluding the original word), in fact, gender/number agreement errors of the same word.

If this amount of evaluations could lead to an accurate representation of the editor's quality or not, remains to be ascertained in the future.

Chapter 5

Conclusions and Future Work

Even though artificial error insertion systems have been made in past, none of them were made with human evaluation as their main goal. For this purpose, we created configurable artificial error insertion tool, aiming to create errors adequate for editor/student evaluation. To accomplish this, ERRORIST is able to introduce errors according to an extensive error taxonomy. By using real error data provided by Unbabel, we fine-tuned the error types for editor evaluation. As there are more error types and methods of introducing any one error type, ERRORIST is extensible by nature, on both error types and their verifiers.

While many error types proved to be generated adequately according to the Costa taxonomy, others proved to be somehow unreliable in their generation.

Despite ERRORIST's shortcomings regarding error quality, it can both be used for automated error insertion and correction detection. It does not replace a human evaluator but it does reduce their work, since 66.83%, $(Ok + Ko)/Total$, of editor corrections can be detected automatically, even further (70.48%) if you consider non-translation editing. Even in a business environment, ERRORIST proved its usefulness by creating adequate errors for editor evaluation and by tracing 58.82% of those errors.

ERRORISTprovides a way to evaluate grammatical skills of anyone who might benefit from doing so. This includes English/Portuguese learners of any nationality (including native English/Portuguese speaking children), publication editors (book editors, magazine editors, etc.) and of course, translation editors.

ERRORIST and all the data gathered in this work will be made available upon publication.

5.1 Future Work

While ERRORIST provides a great variety of errors they are not as reliable as they should be. Error insertion would highly benefit from an additional verification step, where inserted errors would be classified as either visible or invisible. This classification would sanitize the produced errors greatly.

ERRORIST should be able to prioritize some error types over others since some are easier to insert than others. For example Contraction errors have a somewhat limited number of candidates in order to be introduced, in contrast to Omission errors, which any word can be considered as a candidate in order to introduce. Contraction errors should be considered first for insertion as they are more difficult to insert.

ERRORIST's error generation could also be improved in general, as some errors' generation methods are still unreliable regarding their intended error type. Some cases could be improved with some further refinement, like `Confusion of Senses` disregarding articles as candidates for error generation, while others would prove more difficult to improve, like `Misselection: Verb` errors due to requiring a more precise generation method both in verb type identification (regular vs irregular) and entry selection for suffix change.

Error Tracing would greatly benefit from paraphrase detection as adequate synonyms or paraphrases should also be considered a good correction. While this could be achieved somewhat directly in simpler errors, like `Omission` errors, it would require significant research for other errors. The Tracing would also benefit from using an alignment tool as the currently used methods are very rudimentary.

As one of the goals for ERRORIST is its extensibility, a further variety of errors would diversify ERRORIST's error type variety and, therefore, provide a better coverage in editor evaluation.

As of now, ERRORIST supports EP errors as well as some English errors. It could be extended to fully support English and other languages as well.

Bibliography

[1] J. Foster and Ø. Andersen. GenERRate: generating errors for use in grammatical error detection. *NAACL HLT 09 Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, (June):82–90, 2009. URL http://dl.acm.org/citation.cfm?id=1609855.

[2] Â. Costa, W. Ling, T. Luís, R. Correia, and L. Coheur. A linguistically motivated taxonomy for machine translation error analysis. *Machine Translation*, 29(2):127–161, 2015. ISSN 1573-0573. doi: 10.1007/s10590-015-9169-0. URL http://dx.doi.org/10.1007/s10590-015-9169-0.

[3] C. James. *Errors in Language Learning and Use: Exploring Error Analysis*. Longman, University of California, 1998.

[4] V. Mariana, T. Cox, and A. Melby. The multidimensional quality metrics (mqm) framework: a new framework for translation quality assessment. *The Journal of Specialised Translation, (January)*, 2015.

[5] J. Bigert, L. Ericson, and A. Solis. AutoEval and Missplel: two generic tools for automatic evaluation. *NODALIDA 2003 - Nordic Conference of Computational Linguistics*, 3, 2003.

[6] E. Agirre, K. Gojenola, K. Sarasola, and A. Voutilainen. Towards a single proposal in spelling correction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 22–28, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/980845.980850. URL http://dx.doi.org/10.3115/980845.980850.

[7] F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964. ISSN 00010782. doi: 10.1145/363958.363994.

[8] E. Izumi, K. Uchimoto, and H. Isahara. The {NICT} {JLE} Corpus: Exploiting the Language Learners' Speech Database for Research and Education. *International Journal of The Computer, the Internet and Management*, 12(2):119–125, 2004.

[9] J. Sjobergh and O. Knutsson. Faking errors to avoid making errors: machine learning for error detection in writing. *Ranlp*, 2005.

[10] J. Foster. Treebanks gone bad: Parser evaluation and retraining using a treebank of ungrammatical

sentences. *International Journal on Document Analysis and Recognition*, 10(3-4):129–145, 2007. ISSN 14332833. doi: 10.1007/s10032-007-0059-8.

[11] J. Wagner. Judging Grammaticality : Experiments in Sentence Classification. *CALICO Journal*, 26 (3):474–490, 2009. ISSN 0742-7778.

[12] A. Rozovskaya and D. Roth. Training Paradigms for Correcting Errors in Grammar and Usage. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (June):154–162, 2010. URL `http://www.aclweb.org/anthology/N/N10/N10-1018`.

[13] K. Imamura, K. Saito, K. Sadamitsu, and H. Nishikawa. Grammar Error Correction Using Pseudo-Error Sentences and Domain Adaptation. *Acl*, (July):388–392, 2012. URL `http://www.aclweb.org/anthology/P/P12/P12-2076.pdf`.

[14] M. Felice. Generating artificial errors for grammatical error correction. *EACL 2014*, (2006):116–126, 2014.

[15] S. Granger. The International Corpus of Learner English: a new resource for foreign language learning and teaching and second language acquisition research. *Tesol Quarterly*, 37(3):538–546, 2003. ISSN 00398322. doi: 10.2307/3588404. URL `http://onlinelibrary.wiley.com/doi/10.2307/3588404/abstract`.

[16] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999. ISSN 08856125. doi: 10.1023/A:1007662407062.

[17] A. Rozovskaya, M. Sammons, and D. Roth. The UI System in the HOO 2012 Shared Task on Error Correction. *NAACL'12 Workshop on Innovative Use of NLP for Building Educational Applications*, pages 272–280.

[18] D. Dahlmeier, H. Ng, and S. Wu. Building a large annotated corpus of learner English: The NUS corpus of learner English. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, 2013.

[19] F. Sánchez-Martínez and J. A. Pérez-Ortiz. Philipp Koehn, Statistical machine translation. *Machine Translation*, 24:273–278, 2010. ISSN 0922-6567. doi: 10.1007/s10590-010-9083-4.

[20] S. Bird, E. Loper, and E. Klein. *Natural Language Processing with Python.* O'Reilly Media Inc., 2009.

[21] R. Gorin, P. Willisson, W. Buehring, G. Kuenning, et al. Ispell, a free software package for spell checking files. *The UNIX community*, 1971.

[22] Wiktionary. Wiktionary, a collaborative project to produce a free-content multilingual dictionary, 2016. URL `https://en.wiktionary.org/`. Accessed 12-10-2016.

[23] E. Bick, D. Santos, S. Afonso, and R. Marchi. Floresta sintá(c)tica: Ficção ou realidade? *Avaliação conjunta: um novo paradigma no processamento computacional da língua portuguesa. Lisboa, Portugal*, pages 291–300, 2007.

Appendix A

The L2F taxonomy and Invisible errors

Error Type	Lang	Intended Sentence	Visible	Invisible
Punctuation	EN	I found the clowns, Bob, and Clyde.	I found, the clowns, Bob, and Clyde.	I found the clowns, Bob[ ] and Clyde.
	EP	Encontrei os palhaços, João, e Cláudio.	Encontrei os, palhaços, João, e Cláudio.	Encontrei os palhaços, João[ ] e Cláudio.
Capitalization	EN	I think my poor Slipper got dirty!	i think my poor Slipper got dirty!	I think my poor slipper got dirty!
	EP	A minha pobre Pantufa ficou suja!	a minha pobre Pantufa ficou suja!	A minha pobre pantufa ficou suja!
Spelling	EN	I have three friends.	I have htree friends.	I have tree friends.
	EP	Eu fui para a casa.	Eu fiu para a casa.	Eu fui para a caça.

Table A.1: Orthography level invisible error examples

Consider the invisible error in the Punctuation example, the lack of a comma turned 'Bob' and 'Clyde' into clowns. The introduced punctuation error maintains grammatical correctness but changes the sentence's meaning. The same can be said for the Capitalization example, 'Slipper' (or 'Pantufa' in EP) is a cat's name but in the altered sentence those names are now read as their common noun counterparts, changing the sentence's meaning but not their grammatical correctness. In the Spelling example, in English the word 'three' was misspelled and transformed into another existing word 'tree'. In EP, the word 'casa' (house) was misspelled as 'caça' (hunt), changing the sentence's meaning from 'I went to the house.' to 'I went hunting.'

Omitting the function word 'already' in the example, changes the sentence's meaning but not its grammatical correctness. Inversely, adding a function word can create a invisible error just as well.
Omitting the word 'hat' in the example creates a invisible error transforming 'his' from a determinant into a pronoun. Once again, inversely the addition can create an invisible error in the same way.
Untranslated errors creates a invisible error by not translating a word that has an adequate homograph in a target language.

Error Type	Lang	Intended Sentence	Visible	Invisible
Omission (function word)	EN	On his birthday he tried a new hat on.	[ ] his birthday he tried a new hat on.	On his birthday he tried a new hat [ ].
	EP	Ele já recebeu um chapéu no seu aniversário.	Ele já recebeu um chapéu [ ] seu aniversário.	Ele [ ] recebeu um chapéu no seu aniversário.
Addition (function word)	EN	He bought a hat.	He bought a (already) hat.	He bought a hat (already).
	EP	Ele comprou um chapéu.	Ele comprou um (já) chapéu.	Ele (já) comprou um chapéu.
Omission (content word)	EN	His hat was prettier.	His hat was [ ].	His [ ] was prettier
	EP	O seu chapéu era bonito.	O seu chapéu era [ ].	O seu [ ] era bonito.
Addition (content word)	EN	His was prettier.	(suit) His was prettier.	His (suit) was prettier.
	EP	O seu era mais bonito.	(chapéu) O seu era mais bonito.	O seu (chapéu) era mais bonito.
Unstranslated	From EN	Boys like bugs, as girls like dresses.	Boys like bugs, as girls like dresses.	Boys like bugs, as girls like dresses.
	To EP	Os meninos gostam de insectos, como as meninas gostam de vestidos.	Os boys gostam de insectos, como as meninas gostam de vestidos.	Os meninos gostam de insectos, as meninas gostam de vestidos.
	From EP	Os meninos gostam de insectos, as meninas gostam de vestidos.	Os meninos gostam de insectos, as meninas gostam de vestidos.	Os meninos gostam de insectos, as meninas gostam de vestidos.
	To EN	Boys like bugs, girls like dresses.	Boys like insectos, girls like dresses.	Boys like bugs, as girls like dresses.

Table A.2: Lexis level invisible error examples

Error Type	Lang	Intended Sentence	Visible	Invisible
Misselection (word class)	EN	The cute bird is happily on the branch.	The cutely bird is happily on the branch.	The cute bird is happy on the branch.
	EP	O lindo pássaro estava alegremente na árvore.	O lindamente pássaro estava alegremente na árvore.	O lindo pássaro estava alegre na árvore.

Misselection (verb level: tense)	EN	He had bought a suitcase for his travels.	He had buy a suitcase for his travels.	He [ ] bought a suitcase for his travels.
	EP	Ele tinha comprado uma mala para as suas viagens.	Ele ter comprado uma mala para as suas viagens.	Ele comprara uma mala para as suas viagens.
Misselection (verb level: person)	EN	This car is destroyed.	This car am destroyed.	N/A
	EP	Come, porque a viagem é longa!	Come, porque a viagem são longa!	Comei, porque a viagem é longa!.
Misselection (verb level: blend)	EN	They were beautiful yesterday.	They am beautiful yesterday.	N/A
	EP	Ele comeu enquanto pensava nas alturas em que jogaria à bola.	Ele comeriam enquanto pensava nas alturas em que jogaria à bola.	Ele comeu enquanto pensava nas alturas em que jogavam à bola.
Misselection (agreement: gender)	EN	She tied her hair into a knot.	N/A	She tied his hair into a knot.
	EP	Ele atou os cabos do computador por ela.	Ele atou os cabos da computador por ela.	Ele atou os cabos do computador por ele.
Misselection (agreement: number)	EN	The wolf took care of many cubs.	The wolf took care of many cub.	The wolves took care of many cubs.
	EP	O lobo já os tinha alimentado.	Os lobo já os tinha alimentado.	O lobo já o tinha alimentado.
Misselection (agreement: person)	EN	We learn from our mistakes.	N/A	We learn from my mistakes.
	EP	Aprendemos com os nossos erros.	N/A	Aprendemos com os meus errors.
Misselection (agreement: number)	EN	The wolf took care of many cubs.	The wolf took care of many cub.	The wolves took care of many cubs.
	EP	O lobo já os tinha alimentado.	Os lobo já os tinha alimentado.	O lobo já o tinha alimentado.

Misselection (agreement: blend)	EN	All hail our many huntresses in the hunter's guild!	All hail our many hunter in the hunter's guild!	All hail our many huntresses in the huntresses' guild!
	EP	Vou àquela loja de roupa para meninas.	Vou àqueles loja de roupa para meninas.	Vou àquela loja de roupa para menino.
Misselection (contraction)	EN	N/A	N/A	N/A
	EP	Ela adorava sentar-se no banco.	Ela adorava sentar-se em o banco.	N/A
Misordering	EN	I like the beautiful colors on the car.	I beautiful like the [ ] colors on the car.	I like the [ ] colors on the beautiful car.

Table A.3: Grammar level invisible error examples

Error Type	Lang	Intended Sentence	Visible	Invisible
Confusion of senses	EN	The box was full.	N/A	The cashier was full.
	EP	Ele poisou os óculos na mesa.	N/A	Ele poisou os copos na mesa.
Wrong Choice	EN	On New Year's eve I'm going to wear my best suit.	N/A	On New Year's eve I'm going to wear my best truck.
	EP	Na véspera de ano novo vou usar o meu melhor chapéu.	N/A	Na véspera de ano novo vou usar o meu melhor camião.
Collocation	EN	I want to catch the bus and take the pill.	N/A	I want to catch the bus and confiscate the pill.
	EP	Quero apanhar o autocarro e tomar a pílula.	N/A	Quero capturar o autocarro e tomar a pílula.
Idioms	EN	It's raining cats and dogs today!	N/A	It's raining pots today!
	EP	Está a chover a potes hoje!	N/A	Estão a chover cães e gatos hoje!

Table A.4: Semantic level invisible error examples

Error Type	Lang	Intended Sentence	Visible	Invisible
Style	EN	I need permission to be authorized to improvise.	N/A	I need authorization to be authorized to improvise.
	EP	Preciso de autorização para permitir tal loucura.	N/A	Preciso de permissão para permitir tal loucura.
Variety	EN	I'm seeing the most beautiful colors.	I'm seeing the most beautiful colours.	N/A.
	EP	No seu discurso, João...	Em seu discurso, João...	N/A.
Should not be translated	EN	Have you ever read a book written by Fernando Pessoa?	N/A	Have you ever read a book written by Ferdinand Person?.
	EP	Já alguma vez provaste uísque Johnny Walker?	N/A	Já alguma vez provaste uísque João Andante?

Table A.5: Discourse level invisible error examples