ErrorIST 2.0 - Generation of *à la carte* errors

Raquel Cristóvão
raquel.cristovao@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2017

Abstract

The evaluation of learners of a language is a task that requires time and work, and that brings an increase in cost to evaluators. But what if evaluators could automatically generate errors and evaluate learners?

In this document, we present ErrorIST 2.0, errors *à la carte*, a tool capable of generating and inserting errors in texts, and evaluate the human interventions. The implemented tool offers several types of errors – such as syntactic, morphologic, punctuation and spelling errors – that can be adapted to different evaluation contexts, such as evaluation of students or editors.

The system generates errors, by taking into account the needs of the evaluator, and inserts them into texts. After the intervention of the students and/or editors, the system detects the corrections and assigns them a classification.

ErrorIST 2.0 allows the insertion of generic errors and, also, of more language specific errors. Although having Portuguese as the main language, in this thesis, we also tested ErrorIST 2.0 in English and French.

The process of evaluating the corrections automatically is challenging and although ErrorIST 2.0 cannot replace the human intervention, it reduces it in more than 45%, depending on the evaluation scenario.

Keywords: errors, taxonomy, error generator, ErrorIST, error evaluation

1 Introduction

The evaluation of fluents or learners of language is one of the concerns for some projects in the area of Natural Language Processing. With the use of ErrorIST 1.0, a tool developed under the (dos Santos (2016)) master's thesis, it is possible to generate and insert different types of errors in texts, and, by text comparison, automatically evaluate the editor. This document describes a new version of Errorist 1.0, Errorist 2.0, a tool dedicated to generating errors. Given a text, ErrorIST generates a set of errors – syntax, morphology, punctuation, spelling, and semantics – and adds them to the text provided according to the choice of the types of errors made by the evaluator. Once the errors have been introduced, the text is submitted to correction. After the correction has been made, ErrorIST 2.0 evaluates the performance according to the editions made. Our motivation in this work is to develop an improved version of the ErrorIST 1.0, providing new types of errors (e.g common errors, language variety, register, spelling, agreement, prepositions, punctuation), new languages (English and French), and able to adapt the system to the several evaluation scenarios such as students, foreigners, and editors evaluations., avoiding manual error generation and much of correction.

2 Related Work

2.1. Error's Taxonomy

Over the years, was presented different taxonomies of errors like the taxonomies dedicated to human errors presented by Rasmussen (Rasmussen (1982)) and by Pereira (Pereira (1983)). The evaluation of editors is one of the main objectives of this work, which leads us to highlight the Multidimensional Quality Metrics (MQM) (Lommel et al. (2014)) taxonomy and the L2F taxonomy (Costa et al. (2015)).

2.2. Error Generation Software

Most systems use error generation to create training data for use in automatic correction. Felice and Yuan (Felice and Yuan (2014)) have used error generation to correct errors given by English learners as a second language. Based on a corpus annotated with real errors, they injected probabilistic errors

to create corpora to correct the various types of errors. In order to refine the contexts where the errors occur and to insert the errors more precisely, they used a Part-of-Speech (POS) tagger. Finally, the authors concluded that although results vary according to the various training sets, error generation based on actual errors and the use of a POS tagger improves the performance of the correction systems. However, there are some error generating systems such as GenERRate (**?**) and the Missplel (Bigert et al. (2003)), which given a text are capable of generating and inserting several types of errors.

GenERRate is an error generating tool that transforms a well formed sentence into a misspelled sentence. This tool is based on four types of generic operations: insert, delete, replace and move. The insertion of the errors can be done randomly, based on a list of words, and also through the identification of the POS tagger.

Missplel is based too in four basic operations like the GenERRate, but is divided into modules:

- Damerau (Damerau (1964)): introduces errors known as Spelling errors. These errors can be caused by the use of the keyboard.
- Split Compound: creates errors related to compound words.(e.g *"blackboard"* and *"black board"*).
- Sound Error: insert competency [1] errors, that the mistakes are made by not knowing the correct form of writing. It happens when the sounds are similar. (e.g *"bee"* and *"be"*).
- Syntax: inserts agreements errors, verbal errors, repetition errors, and omission errors.

In this system, each error has an associated probability. This allows common misspellings to be introduced more frequently. All errors generated by Missplel are generated randomly, thus giving the user some freedom of choice as to the type of error.

2.3. Error Correction Software

There is a wide variety of grammar checkers, but we highlight the correct one, the Language Tool[2], GNU Aspell[3] and *Grammarly*[4]. For Portuguese, we highlight the Correct (Medeiros (1995)), a checker that uses a morphological parser, the *Palavroso*, that facilitates the process of transformation rules applied to verbs.

Language Tool and Aspell are available for multiple languages and Language Tool contains grammar rules that cover more than 25 languages such as French and Russian.

Grammarly is an automatic grammar checker just available to English. It detects errors of several types and presents corrective suggestions to the user. With the *Grammarly Handbook* it is possible to visualize all the rules applied to the English language.

Contrary to these systems that deal with a very limited type of errors (Missplel focus mainly on spelling errors), or follow a too broad categorisation (GenERRate considers Insertion, Omission, Move, and Substitution errors), ERRORIST follows a finer-grained taxonomy. As a consequence, ErrorIST 2.0 allows the introduction of very specific error types. It allows, thus, to perform a more precise evaluation of the speaker.

3 ErrorIST's Architeture

ErrorIST is composed of five modules: the Error Selector, the Error Generator, the Checker, the Tracer and the Evaluator (Figure 1). In Error Selector the user can select errors of the ErrorIST's 2.0 menu. Then the Error Generator generates the errors, and Checker, optional module, verifies if the word generated exists in the dictionary. The Tracer module analyses how the generated errors were edited and the Evaluator grades the modifications detected by the Tracer. These modules will be described in detail in the following sections.

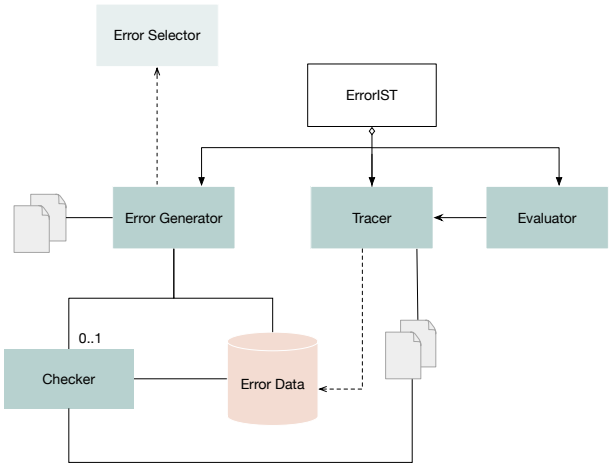

Figure 1: Arquitetura do ErrorIST 2.0

3.1. Error Selector

In this module, the user selects a text and the types and number of errors that will be generated and inserts the files needed to generate the errors corresponding to each language. ErrorIST 2.0 has been inspired by different taxonomies of errors such as L2F and MQM, and in statistical data obtained by experts to provide the following menu of errors:

- Punctuation (omission and addition)(e.g: I found *,* the clowns, Bob, and Clyde.);

[1] https://motivatedgrammar.wordpress.com/tag/competence/
[2] http://wiki.languagetool.org
[3] http://aspell.net
[4] https://www.grammarly.com/faq#toc0

- Capitalization (e.g: *i* think my poor Slipper got dirty!);
- Spelling and Keyboard (e.g: I have *htree* friends.).
- Whitespace(e.g: I love ** bananas.).
- Omission (content, function words, pronouns, determiners or prepositions) (ex: His hat was []);
- Addition (content or function words) (ex: He bought an *already* hat.).
- Misselection
  - Word-class (e.g: The *cutely* bird is on the branch.);
  - Verbs (tense, person or both) (e.g: He had *buy* a suitcase.);
  - Agreement (gender, number or both; or person)(e.g: *Os lobo* fugiu.(The Wolf run away. In Portuguese, Os is plural and lobo singular.));
  - Contraction (e.g: Ela senta-se *em a* cadeira.(She seats in the chair. In Portuguese, em + a = na.)).
- Misordering (e.g: I *beautiful* like the [] color of your eyes.);
- Confusion of senses: when a word was translated into one of its possible meanings, but, in the given context, not the correct one
- Graphic Accentuation (e.g *Catia* is my sister. (Cátia));
- Register: level of formality higher or lower (e.g Se *quiser* eu dou-te. (If you want, I'll give it to you.));
- Wrong Language Variety (e.g telemóvel (EP) *celular* (BP) (cellphone));
- Wrong (determiner and preposition) (e.g I arrived *to* the airport late.);
- Proclise (change the pronoun position)(e.g "Que *chama-se*") (that is called);
- Commons (e.g *expresso*, espresso).

3.2. Error Generator

Error Generator receives a text and the type of errors selected by user on the Error Selector and will generate errors using several resources that help to generate the errors:

- POS tagger from Spacy[5] according to the selected language. In this version we have POS taggers available for Portuguese and English;
- Affix files from Aspell with rules that are applied to change the word suffix according to its category;
- Others input files with lists with handcrafted rules with some language particularities. For instance, spelling rules for Portuguese and French.

The mode of errors generation is different for each type of error. In **Punctuation** errors ErrorIST 2.0 adds and removes commas, semicolons and points. In order to make punctuation errors more likely to be misguided as actual errors, we use the POS tagger to identify the word classes of each sentence. Then, it is added a comma between a noun (or pronoun) and a verb, or omitted a comma after an adverb. **Capitalization** are inserted by changing the initial letter of a noun to uppercase or lowercase. In **Spelling** you can omit, add, replace or change the order of the letters. The user can also add handcraft rules for spelling errors, for instance, a common mistake in **E**uropean **P**ortuguese is to use *'ç'* instead of *'ss'*. **Commons**, **Contraction** and **Graphic Accentuation** errors are inserted too by handcraft rules available by input files. The **Keyboard** errors are spelling errors introduced based on the keyboard and neighboring keys. In **Whitespace** errors added a space between two words. **Misordering** errors randomly change the position of a word in a sentence.

Omission errors are created by removing a word from the sentence, **Addition** errors are created by adding a word to the sentence by proving a file with specific words (to be added). And **Wrong** errors are created by replacing a word per another word of the same category (e.g prepositions). In these errors, we can choose to omit/add a content word (e.g noun) or a function word (e.g conjunction). If we want to be more specific it is possible to omit and change (with input files) prepositions, determiners, and pronoun, with help of POS tagger.

The **Misselection** errors use the POS tagger and affix files allow generating errors such as **Agreement Gender** and **Verb Tense**. For instance, if we introduce an **Agreement Gender** error, in EP, in the word *"menina"* (girl), the Error Generator apply the affix files rules – delete *'a'* and add *'o'* – and we obtained the word *"menino"* (boy). In the **Word Class** errors have selected a word for changing the suffix and his category with affix rules. As an example, the verb, in EP, *"aderir"* (to adhere) into the adjective *"aderente"* (adherent).

[5]https://spacy.io

The Wiktionary is used by **Confusion of Senses** to find translations of the word. It selects a word and searches in a Wiktionary page. Then, it uses the translated word to obtain all of his meanings and returns non-intended meanings. For instance, in EP, the word *"rede"* (net) in the sentence *"Rede social"* (social network) can be translated into *"armadilha"* (trap). And the incorrect sentence is *"Armadilha Social"* (social trap).

There are available specific errors for European Portuguese like the **Proclise** errors that are related to the position of the pronoun. With POS tagger if the sequence conjunction or adverb, personal pronoun and verb are found, the personal pronoun change the position to after verb, and adds a hyphen between verb and pronoun. For example, *"...que me deu."* (what gave me) is transformed into *"...que *deu-me*."*. **Language Variety** errors are available for the variation between **E**uropean **P**ortuguese and **B**razilian **P**ortuguese, and are inserted by lists with handcraft rules (e.g (access) *"aceder"* (EP) *"acessar"* (BP)). Discourse errors are characterized as errors of variation in the text such as speech formality. **Register** errors are a speech formality errors and use POS tagger to identify pronouns, in EP, that preside a verb. For example if exists the word *'me'*, before a verb *"quiseres"* ErrorIST 2.0 change the verb to *"quiser"* like an **Agreement Person** error or **Verb Person**. It inserts one error per sentence and if an error cannot be applied to the current sentence, Error Generator moves to the next sentence.

3.3. Checker

Checker is an optional module that verifies the words in dictionary. For example, when ErrorIST 2.0 generate a **Verb Tense** if the word *"*trazi*"* is generated, the checker will check in the dictionary that it does not exist and the Error Generator repeats the generation process again. The dictionary is a corpus that can be expanded.

3.4. Tracer

The Tracer receives as input a file with the correction made by the editor and identifies the changes made to the file and evaluates if:

- `C(changed)`: C = 1 if the editor edited the error; C = 0 if not.
- `E(xpected)`: E = 1 if the editor made the expected modification in the error into original form; E = 0 if not.
- `O(ther)`: O = 1 if the editor modified the sentence in any other position; O = 0 if not.

As an example, consider the sentence *'"Mary want a bag."*, which was modified by ErrorIST 2.0 into *"*ary* want a bag."*. We can have the following situations:

- The editor modifies the sentence back to *"Mary want a bag."*. In this case, C = E = 1, and O = 0, as the error was corrected as expected, and no other modifications occurred.
- The editor returns *"Mary want a *little* bag."*. Here, C=E=O=1.
- The editor returns *"*Susy* want a bag."*. As a result, C=1 and E=O=0.
- The editor does not modify the sentence at all. In this case, C=E=O=0.

3.5. Evaluator

After the Tracer check and error status evaluation, such as `Change`, `Expected` and `Other`, the Evaluator evaluates the editor's correction as `OK` (correct), `INDEF` (undifined), and `KO` (incorrect). Considering the same phrase *"*ary want a bag."*

Taking the example, the assessment would be (Table 1):
OK - the phrase was modified and the result was expected, as there are no other changes, ie, the editor corrects *"ary"* for *"Mary"*;
INDEF - the phrase has been modified, but not as expected, for example *"Mary want a skirt"* (and a human check is required in these cases);
KO - the sentence with the error has remained unchanged.

Table 1: Evaluator

Eval	C	E	O
OK	1	1	0
INDEF	1	1	1
	1	0	0
	1	0	1
KO	0	–	0

The user can provide a file with the weights associated with each type of error. The assignment of weights to errors will be useful to distinguish between the several types of errors and penalize or benefit the appraised according to their performance.

4 Evaluation

4.1. Error's evaluation

In the first evaluation of the ErrorIST 2.0 it was evaluated if the inserted error corresponded to its objective. We consider that the insertion was successful when, for example, when inserting a **Verb Person** error in the verb, in EP, *"vi"* we get the word *"viu*"*. If the word obtained is *"veste"*, we consider it an unsuccessful insertion, since there is

no time in the verb *"ver"* that has the conjugation of the *"veste"*. For this evaluation were generated 8 errors of each of the types available in the menu, and inserted in journalistic texts. We then asked the 2 experts to evaluate each error according to the expected objective, and was obtained the results of Table 2.

Table 2: ErrorIST Menu

Error	Success	Failure
Agreement Person	8	0
Agreement Gender	1	**7**
Agreement Number	8	0
Addition Content	7	1
Capitalization	5	**3**
Contraction	8	0
Common	8	0
Confusion of Senses	3	**5**
Graphic Accentuation	8	0
Keyboard	8	0
Misordering	8	0
Omission Content	8	0
Omission Determiner	8	0
Omission Pronoun	8	0
Omission Preposition	8	0
Proclise	7	1
Punctuation Omit	8	0
Punctuation Addition	7	1
Register	6	2
Spelling	8	0
Verb Person	6	2
Verb Tense	6	2
Whitespace	8	0
Word Class	5	**3**
Wrong Determiner	8	0
Wrong Prepositon	7	1
Wrong Language Variety	8	0
Resultado	**184**	**32**

Of the 216 errors generated we concluded that the errors were successfully inserted 85.18% of the time. We should point out that there are 14.82% of the errors in which the POS tagger analysis failed or that resulted from the incorrect application of the rules of the affix files.

According to of table, **Agreement Gender** stands out because has a greater number of insuccess cases. When we insert an error into the word *"algumas"*, with the affix files, is generated the word *"*algumos*"*. In this case, the gender of *"algumas"* is *"alguns"*, then is an example of the insuccess case. The same thing occurs with **Word Class** errors that use the same generation mode. The suffix is altered and category too. Verifying the existence of words generated using affix files was an issue in the ErrorIST version 1.0. Thus, in order to solve cases of failure such as the Agreement Gender Errors and the Word Class Errors, we activated the Checker module and re-generated 8 errors of each type and got the results from the table 3.

Table 3: Results with Checker

Error	Success	Failure
Agreement Gender	6	**2**
Word Class	8	**0**
Result	**14**	**2**

With the active checker we were able to reduce the number of cases of failure in both errors.

4.2. Evaluation with Portuguese students

For this evaluation, it was necessary to understand that were the more frequent errors produced by students in a university context. In the discipline of Writing, in general, not only do they analyze and produce different textual genres, but also evaluate the textualizations produced. In order to better understand the scope of ErrorIST 2.0, inserting automatic errors and evaluating their expression frequently, in the ecological context of the classroom is thus essential. Taking into account this objective, we tried to understand which errors were more frequent. The study was carried out over 4 semesters and it was concluded that the most frequent types of errors, of the students, are the errors of syntax, punctuation, spelling and morfossintaxe.

The evaluation was done with 14 students and the errors from ErrorIST 2.0 that best fit the type of sentences to be evaluated, according to the teachers, were the **Commons, Graphic Accentuation, Proclise** errors and **Punctuation Omit**. For each type of error 10 errors were generated and inserted in journalistic texts. From the successful inserts, presented in Table 5.1, the teachers selected the errors that were appropriate for the students.

From the successful inserts, presented in Table 2, the teachers selected the errors that were appropriate for the students. They selected 17 errors from the following classes of errors: **Common, Graphic Accentuation, Proclise** and **Punctuation Omit**. The texts were given to students for correction without any information about error's type inserted. After the edits are complete, the ErrorIST 2.0 evaluated the students according to the Evaluator's evaluation metrics (Subsection 3.5).

In the Table 4, we can observe the results from ErrorIST 2.0 with Number of insertions, corrected errors (`OK`), incorrect errors (`KO`), undefined (`INDEF`), and the total discarded errors (`T = (OK + KO)/N*100`). In `INDEF` cases ErrorIST 2.0 cannot evaluate as `OK` or `KO` and `INDEF` cases require human verification. One of the experts did the ver-

Table 4: Evaluation with portuguese students

Type	N	OK	KO	INDEF	%T
Common	56	18	6	32	42,85%
Accent.	42	11	7	24	75%
Proclise	28	13	3	12	57,14%
Punct. Omit	112	38	12	62	44,64%
Total	**238**	**80**	**28**	**130**	**45,38**

ification and obtained the following results:

Table 5: Verification of undefined cases

Type	N	OK	KO
Common	32	6	26
Accent.	24	3	21
Proclise	12	2	10
Punct. Omit	62	33	29
Total	**130**	**44**	**86**

One of the motivations of ErrorIST 2.0 is the reduction of the number of human interventions and on 45,38% of interventions were avoided. The high number of undefined cases may be influenced by the type of error. In **Punctuation** errors, human verification is necessary, when a comma is inserted beyond the one desired. The addition of a comma can keep the sentence correct.

4.3. Evaluation with French speakers

We selected French as the new language for ErrorIST 2.0, and we evaluated with 3 French speakers with a different degree of schooling.

Firstly, we prepared the evaluation using common language errors in the Language Tool, and other error study documents in French Strube Den Lima (1990). After this study, the expert selected as errors: **Common, Contraction, Graphic Accentuation**, and **Spelling** errors for the evaluation. Then, ErrorIST 2.0 generated 10 errors of each type, and the expert chose only 14 out of 40 errors, that was introduced in journalistic texts. The selected errors are not dependent on affix files and POS tagger so the insert mode was successful in all cases.

The text was given to fluent speaker in French to correct it, and without any information about the types of errors; they were only informed that there was one error per sentence. Once again, after being edited by a fluent, ErrorIST 2.0 made the classification of the editions, presented in Table 6.

Taking into account that the evaluation was simpler and with only 3 participants, the number of undefined cases were smaller. Analyzing the results, it is concluded that ErrorIST 2.0 was able to evaluate 76.19% of errors as correct or incorrect. The expert identified 10 undefined cases and concluded that just 2 cases were correct.

Table 6: Evaluation of French speakers by ErrorIST 2.0

Type	N	OK	KO	INDEF	%T
Common	18	14	1	3	83,33%
Contract.	6	2	1	3	50%
Accent.	12	9	1	2	83,33%
Spelling	6	4	0	2	66,67%
Total	**42**	**29**	**3**	**10**	**76,19%**

One of the undefined cases was in a **Contraction Error**, where the evaluated ones did not detect the error and changed the sentence context without editing the error as expected. In the original sentence *"Je vais à la pharmacie." (I am going to the pharmacy.)* was insert the **"Contraction Error"** and obtained the sentence *"Je vais* ***au*** *pharmacie." (I go to the pharmacy.)*, and the fluent speaker in French corrected to *"Je vais au* ***gym****." (I'm going to the gym.)*. In this cases, it is necessary a human verification.

4.4. Evaluation With Unbabel

The evaluation of editors is a costly task for Unbabel. As a proposed solution to the problem, we have analyzed the most frequent types of errors of the editors together with an Unbabel expert and added new types of errors to ErrorIST 2.0. Were added to ErrorIST new errors such as **Wrong Preposition, Wrong Determiner, Omission Preposition, Omission Determiner, Wrong Language Variety, Register, Whitespace** and **Graphic Accentuation**.

For the evaluation with editors were generated 10 errors of each type, in EP, available in the menu of ErrorIST 2.0 and inserted them in texts with e-mail format. From the set of errors generated were selected the most suitable for the texts, as depicted in Figure 2.

Although, we have not been able to get the results of errors that are being validated by Unbabel experts. After this validation, the objective of our work would be to complete the evaluation with the interaction of Unbabel's editors and evaluate ErrorIST's utility in this context.

5 Discussion

After the evaluation and observing the previous tables, we can conclude that ErroIST 2.0 was able to evaluate as correct or incorrect 45.38 % of errors in the evaluation with students and 76.19 % in the evaluation with fluent speaker in French

In the evaluation with Portuguese students the

Common errors and the **Punctuation** errors were notable for having an unexpected number of cases greater than 50%.

In the case of **Punctuation** errors the number of unexpected cases, in percentage, is greater than the presented by **Common** errors. ErrorIST 2.0 does not contain a system of syntax disambiguation to identify verbal and noun phrases, in Punctuation errors, and can not identify cases where commas were added or omitted correctly by the evaluated one, eventually returning undefined.

The percentage of unexpected cases in the evaluation with foreigners was lower (23.81 %). The provision of ErrorIST 2.0 allowed only 10 errors (in 42) to be manually checked.

In Figure 2 we can see the menu of errors made available by ErrorIST 2.0 and the errors selected for each one of the evaluations performed by ErrorIST 2.0. The error menu of ErrorIST 2.0 allows you to adapt the system to different evaluation scenarios taking into account the needs presented by the experts.

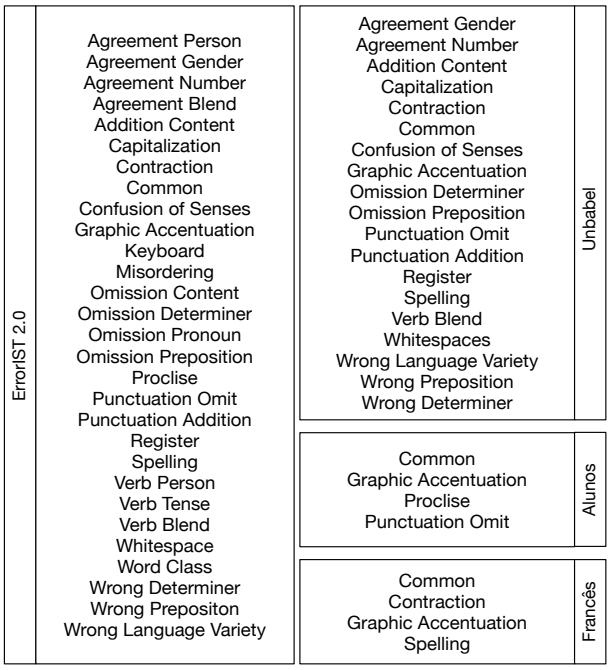

Figure 2: ErrorIST Menu and errors selected for each context

6 Conclusions

ErrorIST 2.0 shows improvements over ErrorIST version 1.0, previously presented. ErrorIST 2.0 was aimed at: improving the way of generating capitalization errors, punctuation and verbs conjugations; increasing the number of errors made available; introducing speech errors; introducing new languages; evaluating the system in new scenarios.

In **Capitalization** errors, the capitalization was made only in words whose POS tagger identifies as nouns. ***Punctuation*** errors inserted any type of character and were limited to the main punctuation characters. To solve the generation of non-existent words, a verification module was added to the system architecture, the module *Checker*, which can be activated if the user so desires. The introduction of the *Checker* module decreased the number of failures of inserting errors which use the affix files in their generation, such as ***Verb*** errors, ***Agreement*** errors and ***Word Class*** errors.

In comparison to GeneRRate and Missplel generation software, and to the previous version, ErrorIST 2.0 provides a more complete error adaptable to different types of evaluation. The menu provides generic errors, which do not require auxiliary resources, and also more specific errors of a given language. In addition to errors in Portuguese, ErrorIST 2.0 provides some types of errors for English and French. For the introduction of new error types in ErrorIST 2.0, statistical data provided by experts have been taken into account in each one of the evaluation – with fluent Portuguese speakers, fluent French speakers and Unbabel editors – scenarios.

The main goal of ErrorIST 2.0 is to minimize the task of human validation and, despite the high number of undefined cases, ErrorIST 2.0 was able to decrease the manual task by more than 45% of errors. In the evaluation with Portuguese-speaking students, ErrorIST 2.0 exempted 45.38% of the errors from manual verification and in the evaluation with French sources it was able to automatically evaluate 76.19% of errors. These results have several factors involved as the types of errors generated, the level of knowledge of the evaluated ones and also the verification made by ErrorIST 2.0, which may be improved in the future.

With the improvements made in the version ErrorIST 2.0, we conclude that with the menu of errors provided by ErrorIST 2.0, in Figure 2 it is possible to adapt the system to different assessment scenarios according to the needs of the experts.

References

J. Bigert, L. Ericson, and A. Solis. Missplel and autoeval: Two generic tools for automatic evaluation. In *Proceedings of Nodalida*. Nordic Conference of Computational Linguistics, 2003.

Â. Costa, W. Ling, T. Luís, R. Correia, and L. Coheur. A linguistically motivated taxonomy for machine translation error analysis. *Machine Translation*, 2015.

F. J. Damerau. A technique for computer detection

and correction of spelling errors. *Commun. ACM*, 1964.

T. M. F. dos Santos. Errorist: towards the automatic evaluation of editors. *Master's thesis, Instituto Superior Técnico-Universidade Técnica de Lisboa*, 2016.

M. Felice and Z. Yuan. Generating artificial errors for grammatical error correction. In *EACL*, 2014.

A. Lommel, H. Uszkoreit, and A. Burchardt. Multidimensional quality metrics (mqm): A framework for declaring and describing translation quality metrics. *Tradumàtica*, 2014.

J. C. Medeiros. Processamento morfológico e correcção ortográfica do português. *Master's thesis, Instituto Superior Técnico-Universidade Técnica de Lisboa, February*, 1995.

O. G. Pereira. Erro humano: uma conferência internacional. *Análise psicológica*, 1983.

J. Rasmussen. Human errors. a taxonomy for describing human malfunction in industrial installations. *Journal of occupational accidents*, 1982.

V. L. Strube Den Lima. *A contribution to the study of error treatment at lexical-syntactical level in a french written text.* PhD thesis, Mar 1990.