Corpus-based Evaluation of a French Spelling and Grammar Checker

Marianne Starlander and Andrei Popescu-Belis

ISSCO/TIM, University of Geneva,
École de Traduction et d'Interprétation
40 Bvd. du Pont d'Arve
CH-1211 Geneva 4 – Switzerland
marianne.starlander@eti.unige.ch
andrei.popescu-belis@issco.unige.ch

Abstract

This article describes an evaluation method for spelling and grammar checkers and gives the results of its application to two French checkers. The evaluation process follows closely the ISO/IEC and EAGLES guidelines, and defines precisely the evaluation metrics, so that they can be easily reproduced. The choice of professional translators as user profile entails the use of a corpus of spelling mistakes, which was collected and annotated. The metrics are divided into three sets: classification of perfect vs. imperfect sentences; detection of mistakes; correction of mistakes. The results show in which respect the two systems are the most adapted to the user needs, and the points on which they could be improved.

1. Introduction

The methodology of evaluation in Natural Language Processing (NLP) has witnessed considerable standardization efforts in the past decade. Even if programs addressing complex NLP tasks are still hard to evaluate, lower-level programs can now be evaluated following a standard methodology, which enables one to tailor evaluation to the intended context of use, as well as compare results through different evaluations.

The present article has two objectives. First, we aim at applying a standard evaluation process to *spelling and grammar checker* (SGC) programs, producing an evaluation specification that will help future evaluations. Given the increasing number of spelling and grammar checkers, it is indeed necessary to evaluate them impartially. Second, we show how the evaluation process is adapted to a specific type of user, namely a professional translator, in particular through the collection of a corpus of spelling mistakes. Two existing systems for French are evaluated and compared from the point of view of such a user.

The article is organized as follows. The rest of section 1 situates our evaluation in the framework of several standards (ISO/IEC, EAGLES) and previous spell checker evaluations (TEMAA). Section 2 outlines the first stage in the evaluation process, namely the definition of the quality requirements, based on the user profile and the description of the corpus and its annotation. Section 3 completely specifies the evaluation metrics, so that they can be reused by other evaluators. Section 4 provides the evaluation results and discussion of them, before the final conclusion (Section 5).

1.1. Principles of Evaluation: Generic Software, NLP Software, Spelling Checkers

The International Organization for Standardization (ISO), together with the International Electrotechnical Commission (IEC) have defined several standards for software evaluation, in two series: ISO/IEC 9126 (1 to 3) and ISO/IEC 14958 (1 to 6), some being still in press. We cannot summarize here the numerous aspects that are dealt with by these standards, but the first series is concerned with quality models, and the second with the evaluation process. Most important for us is that, according to ISO/IEC 14598-5 (1998), the evaluation process is divided into five stages:

- establishment of the quality requirements (the list of required quality characteristics);
- specification of the evaluation (mapping measurements to requirements);
- design of the evaluation (documenting procedures used to perform measurements);
- execution of the evaluation;
- conclusion of the evaluation (and reporting).

Regarding the quality characteristics, ISO/IEC 9126-1 (2001) states that they must belong to the following six top-level classes: functionality, reliability, usability, efficiency, maintainability, and portability.

The European EAGLES initiative aimed at developing standards for NLP software evaluation (EAGLES (Expert Advisory Group on Language Engineering Standards : Evaluation Working Group, 1993-1999). There was strong commitment towards applying the ISO/IEC standards to NLP (ISO/IEC 9126 was first published in 1991), and also towards giving a central place to the *user* throughout the evaluation design. The two phases of EAGLES produced guidelines for NLP software evaluation (cf. the " EAGLES Seven Step Recipe", http://www.issco.unige.ch/projects/eagles/ewg99/7steps.html), including the definition of a quality model, then studied their application to case studies: spell checkers (cf. TEMAA) and machine translation (in the ISLE follow-up project).

1.2. Our Contribution to the Methodology of Spelling and Grammar Checker Evaluation

Given the existing evaluation initiatives mentioned above, our goal was to design an evaluation that conforms to the ISO/IEC and EAGLES guidelines, and apply it to a concrete user profile and associated test data. The evaluation process follows closely the ISO/IEC stages, so the choices made at each point are easily justified, and can be reproduced or modified by others.

With respect to the TEMAA project, which already dealt with authoring assistant tools, our contribution is innovative with respect to at least two points. First, the choice of a human translator as the user model, while still general enough, lead us to use as test data a *corpus of spelling mistakes*, and no longer a test suite that was artificially built. As discussed in Section 2, we believe that this choice brings us closer to capturing user needs. Second, we define our metrics transparently, using formulae, whereas it seems no directly applicable formulae were ever published for TEMAA. We hope thus to provide ready-made metrics for use by future evaluators, and in particular by non professional evaluators who are interested in setting up their own evaluation process.

These innovations are applied to the evaluation of two spelling and grammar checkers for texts written in French. Since these are commercial products (their names will not be disclosed), we had to design a black-box type evaluation (as opposed to glass-box), which relies mainly on test data. The numeric results enable us to draw eloquent conclusions about the adequacy of these tools with respect to our user profile, as well as point out at not-so-performant features. A more detailed version of these results can be found in (Starlander, 2002).

2. Quality Requirements: User Profile and Test Corpus

2.1. Description of the User

As we concentrated on a specific user population, translators, it is important to describe them by setting out their particularities:

- Translators produce very varied texts, style and register as well as content wise.
- Although they usually possess a good command of their working language, they are still likely to make mechanical mistakes, due for instance to copying and pasting text or to typing errors.
- They usually produce a "thought of" text, as the translation process in itself needs some thinking. This implies that the text is not spontaneously produced, as in the case of emails.
- They use a SGC if they can rely on it, meaning that the system is able to detect errors and to correct them, although the second requirement is less important as translators are language skilled enough to find the adequate correction if the mistake is being pointed out.
- Finally, translators, being language professionals, are less tolerant to false alarms than other users, so they would dismiss the use of a weak SGC as time-consuming.

2.2. A Corpus of Spelling Mistakes

2.2.1. Corpus vs. Test Suites

Facing the question of whether to use test suites or a corpus as test material, we decided to choose the latter because we wanted the test material to be representative of a particular type of user. Test suites are best suited to test specific aspects, for example to find out how a system would treat a specific error type such as the past participle agreement. Furthermore, with respect to test suites, corpora take less time to build, and account more precisely for the actual frequency of mistakes, provided they have a reasonable size.

2.2.2. Collecting the Corpus

As we decided to build a user-specific corpora, we collected a corpus of 27 translations (1-3 pages long), produced by French mother tongue students at the Translation and Interpretation School of the University of Geneva. These were written as training exercises, during the academic year 2001.

Most translated texts are general topic articles from the German, English, Italian or Spanish press.

The students were told not to correct their text while producing their translation and not to reread it before sending it. We guaranteed the students a total anonymity in order to keep the mistakes as close to a real situation as possible.

Statistically speaking, the corpus counts a total of approximatively 15,000 words, which is equivalent to around 30 typed pages. The total number of sentences is 637. The corpus is indeed not very large, but it is targeted towards a special population, and we believe it is representative enough for our evaluation. Furthermore, the reduced size allows us to process the corpus manually, as we will see now.

2.3. Annotation of Spelling Mistakes

The raw corpus has been annotated in order to reflect accurately all the mistakes present in it. This implies a human correction, detecting all the errors and subsequently marking them using predefined tags (see below).

In a second phase, the same corpus has been submitted to each SGC system and, using the same tags, we marked the detected mistakes and encoded the suggested correction. In order to encode certain information such as the ranking of the suggestions for correction, we introduced slight changes to the tags used (e.g. we added a number beside the tag reflecting the position of the right correction suggestion).

In general, the annotating of the text should ideally follow encoding directives such as TEI (Text Encoding Initiative) or CES (Corpus Encoding Standard) in order to be reused for other purposes.

Let us see now how the tags were chosen and designed following an error typology which we are now going to describe.

2.3.1. Typology of the Errors

Classifying errors is sometimes difficult as the same error could be classified in various categories; for example in French a missing "s" at the end of a word could be tagged as a typographic error due to omission or as a syntactic one, being caused by a lack of agreement between a noun and its determinant. Thus different types of classification are possible: according to the origin of the error, complexity of error (real word error, long distance error) or type of techniques required to treat them.

We chose a quite classical typology, first of all differentiating between lexical and syntactical errors, and then dividing the first into typographic and phonetic errors. Under typographic errors, the four classical transformation are to be found (Kukich 1992): insertion,

omission, substitution and transposition, as well as two more difficult types: splitting and agglutinating words; and finally the French specific problem of accentuation.

Under syntactic errors, we classified all agreement mistakes (determinant-noun; adjective-noun, subject-verb and past participle), usage errors (such as confusion between past participle and infinitive, which could also be counted in with the phonetically errors., or confusion between, *là* (location adverb) and *la* (determinant)) and construction errors (insertion or omission of a word). These errors are all context dependent.

2.3.2. Resulting tags

The typology developed by Vandenventer et al. inspired us for the choice of the tags themselves. We will briefly describe them below, as well as providing information about the frequency of each type of mistake in our corpus.

	Tag	Description	Freq.
Lexical errors	TOM	Omission of a letter	15.1%
	TIN	Insertion of a letter	13.8%
	TSU	Substitution bet. 2 letters	19.1%
	TTR	Transposition bet. 2 letters	6.9%
	TAC	Accentuation	4.8%
	TCO	Omitting a space bet. 2 words	1.3%
	TSO	Inserting a space bet. 2 words	3.4%
	TTRANS	Transposition oral-written	1.1%
Lexical errors average frequency			67.9%
Syntax errors	SADN	Determinant – noun agreement	10.3%
	SAAN	Adjective – noun agreement	5.3%
	SASV	Subject – verb agreement	1.6%
	SAPP	Past participle agreement	2.4%
	ECON	Usage confusion	7.7%
	CIM	Insertion of a word	0.5%
	COM	Omission of a word	2.4%
	CTRM	Word Order	0.3%
Syntax errors average frequency			32.1%

Table 1. Frequency of various errors in our corpus.

In our corpus, typology errors are more vastly represented (67.9%) than grammar errors (32%). This is probably due to our user profile: as we described above, translators carefully think of the text to be written before typing it. In general, our corpus counts only 377 errors on a total of approximatively 15'000 words. Some error types are poorly represented, such as missing and inserted words, or subject to verb agreements, but this is the result of using a user representative corpus.

2.4. Consequences for Quality Requirements

The definition of a user profile, including user needs, entails the choice of a quality model, that is, a set of quality characteristics that are part of the ISO/IEC 9126 hierarchy. The top-level qualities are: functionality, reliability, usability, efficiency, maintainability, and portability. Our user has advanced knowledge of the writing language, reasonable IT skills, and a need for speed in the production of texts.

Functionality is the main feature in our quality model, i.e., the capacity of the system to detect and correct spelling and grammar mistakes. ISO/IEC 9126 decomposes functionality into suitability, interoperability, security, compliance and accuracy. We have focused our model on the latter characteristic. Indeed, regarding adequacy, it seems quite obvious that the translator needs a checker, and not a language teaching software for instance. The integration into an editor could be studied at length, but many programs offer here similar capacities. Safety and conformance are less applicable here.

We must now decompose accuracy into quality attributes that can be measured (see metrics in Section 3). Two sub-characteristics seem quite obvious, the capacity to detect mistakes and the capacity to suggest corrections, the first one being more relevant to our user type.

Detection can be further divided into the capacity to detect a maximal number of mistakes, and the capacity to avoid false alarms. If the checker raises many false alarms, the user spends a lot of time checking them; if it misses many mistakes, the resulting text is unacceptable. In both cases the user would prefer manual checking. These two capacities lead straightforwardly to recall and precision measures (Section 3).

Correction capacities are also relevant to the user, if they are often accurate, especially since checkers often propose an ergonomic replacement capacity (mouse clicks). The present of the correct suggestion among the proposed ones is here the main quality; preferably, the correct suggestion should be the first one. Even better is the capacity to explain mistakes and provide reasons for correction (but this will not be measured here).

The other five top-level qualities were not used in our evaluation. Efficiency (the link between used resources and provided service) is not relevant since the evaluated software runs fast enough on standard PCs. Of course, the checker's efficiency must not be confused with the user's efficiency in their task, which depends partly on the checker's accuracy but is best measured by quality in use metrics. The other four qualities, reliability, ease of use, maintainability, and portability, are more technical and have not been considered relevant to our user needs. Crashes are infrequent, platforms change too, and the integration of the checkers into common editing applications guarantees ease of use (though ease of installation may vary). If maintainability is related to dictionary updating, it may be relevant to our user, but its measure would imply ergonomic analyses that were beyond our purpose.

3. Specification and Design of the Evaluation

In order to quantify the functionality of a system as an acceptable spelling and grammar checker (SGC), we have defined a number of measures that grasp various aspects of functionality. It is of course not possible to 'prove' that these measures are indeed the right ones. Instead, we describe them fully below, so that the community of evaluators may study them and either adopt them or suggest modifications.

As stated above, our test data is a text corpus containing various spelling and grammar mistakes that were annotated. After giving it the test data, the checker

hypothesizes a certain number of mistakes: some of the guesses are correct (they correspond to real mistakes) and some are wrong (the text was correct in that place). Moreover, we are also interested in the suggestions for correction that were provided for the correct guesses—those for the wrong ones are not relevant. Another measure concerns the ability to validate whole sentences, i.e. to detect sentences containing at least one mistake from those containing none.

We will specify below these three metrics, starting with the last one. For each of them, we define first what exactly is measured or counted, then how the score is computed from these values. The metrics do not necessarily require manual counting, once the mistake corpus is fully annotated; neither do they require particular functions from the checker other than those described above.

3.1. Sentence Level Precision and Recall

3.1.1. How to Count Errors

The first metric determines the system's capacity to correctly classify correct *vs.* wrong sentences. A 'correct sentence' is a sentence containing no spelling or grammar mistakes, while a 'wrong sentence' contains at least one mistake. This capacity helps reducing the revision time for a document, since a human translator is quite good at detecting mistakes when shown in which sentence they are. That is why we introduced the present metric.

The *sentence level metric* involves four quantities that are traditionally known as:

- true positives (*TP*): correct sentences that are marked 'correct' by the checker;
- false positives (*FP)*: wrong sentences that are marked 'correct' by the checker;
- false negatives (*FN*): correct sentences that are marked 'wrong' by the checker;
- true negatives (*TN*) : wrong sentences that are marked 'wrong' by the checker.

3.1.2. How to Compute the Scores

There are several traditional scores used to compute the accuracy of this kind of classifier. If one is interested in the capacity to detect correct sentences, one can compute recall and precision for this task. These are defined as following:

$$R_c = \frac{TP}{TP + FN} \text{ and } P_c = \frac{TP}{TP + FP}$$

Conversely, if one is interested in the capacity to detect wrong sentences, one can compute other recall and precision scores:

$$R_f = \frac{TN}{TN + FP} \text{ and } P_f = \frac{TN}{TN + FN}$$

It is interesting to note that these four quantities are not completely independent, because *TP+FP = FN+TN*. However, it is not completely informative to provide only two of them. Therefore, another score is sometimes used in addition to either two of them, namely predictive accuracy:

$$PA = \frac{TP + TN}{TP + FP + TN + FN}$$

In addition, one can compute f-measure, i.e. the harmonic mean between recall and precision, either for correct sentences or for wrong ones:

$$fm_c = \frac{2}{1/R_c + 1/P_c} \text{ and } fm_p = \frac{2}{1/R_p + 1/P_p}$$

As usual, recall and precision vary between 0 and 1, the best score being one. Recall is the capacity to detect all correct (resp. wrong) sentences, precision is the capacity to detect only correct (resp. wrong) sentences. Trivial strategies enable a system to obtain high recall ('select all the sentences as correct') or high precision ('select very few sentences'), therefore f-measure combines these two scores. The results of our evaluation will take into consideration the four precision and recall scores R_c, P_c, R_f and P_f.

3.2. Precision and Recall for Error Detection

3.2.1. How to Count Errors

The second metric (or rather set of metrics) concerns the detection of the mistakes themselves. More precisely, the system has correctly detected a mistake if it signals a mistake *on the same word*. An exception is made for agreement errors, when signaling either of the two elements that do not agree (e.g., subject + verb) is acceptable, provided there are no other cues (e.g., an adjective or determiner). This task is similar to a retrieval task, and it is quite normal to count, for each type of mistake defined in Section 2, the number of real mistakes found by the checker (*x*), the number of real mistakes not found (*y*) and the number of mistakes hypothesized by the checker that do not correspond to real mistakes (*z*).

3.2.2. How to Compute the Scores

Recall and precision scores are therefore computed for this task, using the above numbers, for each type of mistakes. For 'TOM' mistakes (letter omission) we have :

$$R_{TOM} = \frac{x_{TOM}}{x_{TOM} + y_{TOM}} \text{ and } P_c = \frac{x_{TOM}}{x_{TOM} + z_{TOM}}$$

Once again, the f-measure can be computed as above.

To integrate the scores per type of mistake, it would be unfair to average recall and precision scores, since this would mean that all types of mistakes have the same frequency in the collected corpus. To account for their relative importance, the number of mistakes must be summed up, then the formulae applied, such as, for instance, for recall:

$$R_{\text{detect}} = \frac{\sum_{types} x_t}{\sum_{types} x_t + \sum_{types} y_t}$$

3.3. Precision and Recall for Error Correction

3.3.1. How to Count Errors and Compute Scores

The capacity of the program to suggest the right correction for a detected mistake must finally be measured. It seems fair to take here into account only the suggestions concerning mistakes that were correctly detected. Otherwise, for wrongly hypothesized mistakes, the suggestion would be necessarily wrong (because the text is correct) and this would penalize the checker twice (once for the previous score and once here).

Therefore, for each 'true mistake', we check whether the suggested solution is correct, or wrong, or absent (no suggestion), then total these three figures. Given that our user is probably able to detect the correct suggestion

among a small set of suggestions, we count as a correct suggestion the case when the correct one is present among the set of suggestions (in case there are several), but not necessarily in the first position.

3.3.2. How to Compute Scores

The scores are then quite straightforward to compute: percentage of right suggestions (BS), percentage of false suggestions (FS) and percentage of absent suggestions (AS).

3.4. Integration of the scores

The three sets of metrics defined above provide a multitude of scores. The first set yields (here) two recall/precision couples, one for correct sentences, one for wrong ones. The second yields one recall/precision couple per type of error. The third one yields three percentages of suggestions (correct, false, absent) per type of error. Should all these scores be integrated to provide a single number ?

We believe that reducing all scores to one is not particularly eloquent, even for comparing two systems as in the present case. It seems more relevant to analyze each of the three sets separately, draw conclusions, then integrate these conclusions. Therefore, we will provide below detailed scores for the three sets, and integrate each set separately—that is, average scores across all types of errors.

4. Results

4.1. Execution of the Evaluation

There is nothing much to say about the execution of the evaluation. In our particular case, there was no API available to the two checkers, therefore the results had to be collected manually. The corpus of mistakes was submitted to each of the checkers (without the markup, of course), and the responses were manually annotated on the corpus using similar markup. The correctness of the suggestion was also annotated manually from the graphical response of each checker. The various numbers defined above were then counted, by extracting the tagged portions into tables. The scores were then computed using the above definitions. We provide them hereafter, referring to the two checkers as S1 and S2.

4.2. Scores for the Three Measures

We are going to present the results for detection followed by those for correction.

4.2.1. Sentence Level

	Recall		**Precision**	
	Correct s.	Wrong s.	Correct s.	Wrong s.
S1	87.1%	89.5%	94.4%	77.4%
S2	70.2%	86.4%	91.1%	59.5%

Table 2. Precision and recall for the detection at sentence level (system S1 *vs.* S2).

At sentence level, S2 seems to perform worse than S1, but remember that this measure is only very indicative of accuracy, as sentences marked as incorrect can include varying amounts of errors. For S2, the recall for wrong sentences is quite similar to the one obtained by S1 but the precision rate varies considerably. According to these figures, S1 and S2 are quite comparable on the recall level, but S2 appears to be far less accurate, as S2 counts almost 17 points less for precision on wrong sentences. It will be interesting to see if these first impressions are going to be confirmed by the precision and recall rates on error level.

4.2.2. Error Detection

Here are the results by category of error, first for lexical errors and second for syntax errors:

Lexical Errors				
Error types	**System 1**		**System 2**	
	Recall	**Precision**	**R**	**P**
TOM	78.9%	95.7%	82.5%	95.9%
TIN	76.9%	95.2%	88.5%	80.7%
TSU	79.2%	100.0%	88.9%	98.5%
TTR	84.6%	100.0%	84.6%	100.0%
TAC	84.2%	94.1%	83.3%	88.2%
TCO	100.0%	100.0%	40.0%	66.7%
TSO	80.0%	100.0%	100.0%	100.0%
TTRANS	100.0%	100.0%	100.0%	100.0%
Total	80.2%	97.5%	86.2%	92.6%

Table 3. Recall and precision for error detection (per category of error): lexical errors.

S1 performs quite well on lexical errors, with an average recall of 80.2% and a precision of 97,5%. The difference between these two figures is quite high. Compared to that, S2, although showing lower figures, is far more balanced. Generally speaking, the scores are quite high, as they are all are above 75%, there is not one category of errors scoring particularly low.

Syntax Errors				
Error types	System 1		System 2	
	Recall	Precision	**R**	**P**
SADN	87.2%	81.0%	97.4%	92.7%
SAAN	65.0%	40.6%	90.0%	36.7%
SASV	66.7%	33.3%	66.7%	20.0%
SAPP	11.1%	11.1%	88.9%	72.7%
ECON	34.5%	27.8%	34.5%	22.2%
CIM	0.0%	0.0%	100.0%	100.0%
COM	11.1%	25.0%	55.6%	62.5%
CTRM	0.0%	0.0%	0.0%	0.0%
Total	54.8%	46.7%	73.9%	48.3%

Table 4. Recall and precision for error detection (per category of error): syntax errors.

On the syntax level, the differences between the types of errors are much more significant: the past participle agreement is very low for S1, whereas S2 performs really well. S2 has particularly a high figure for recall for adjective to noun agreement (SAAN), but this has to be tempered by the particularly low precision rate (36.7%).

4.2.3. Error Correction

In the table bellow, we reproduced only the rate of good correction, that is, when the system, once a mistake detected was able to propose the right correction.

	S 1	S 2
TOM	72.7%	74.0%
TIN	75.6%	75.5%
TSU	51.7%	53.0%
TTR	91.3%	72.7%
TAC	100.0%	94.4%
TCO	80.0%	0.0%
TSO	83.3%	15.4%
TTRANS	100.0%	75.0%
Average	72.6%	65.6%
SADN	87.9%	100.0%
SAAN	78.6%	100.0%
SASV	100.0%	100.0%
SAPP	100.0%	85.7%
ECON	85.7%	91.7%
CIM	0.0%	0.0%
COM	100.0%	100.0%
CTRM	0.0%	0.0%
Average	87.0%	69.2%
Total	77.0%	73.9%

Table 5. Percentage of good correction per category of errors for S1 and S2

On average, S1 performs better both on lexical and syntax errors than S2.

The error types which are corrected worse by S2 are the pasting and cutting of two words (TSO and TCO), while S1's worse performances are letter substitution errors (TSU).

The scores are astonishingly high for S2 on syntax errors due to agreement (100%) except past participle agreement presenting a lower rate (SAPP, 85.7%). On the contrary, SAPP has the highest score for S1.

The scores for errors of usage which were really low on detection level, are much better on correction level for both systems. This means, that once such an error is detected, it is well corrected, although this type of error often implies semantic elements.

4.2.4. Overall scores

The overall scores are given here, in the two following tables.

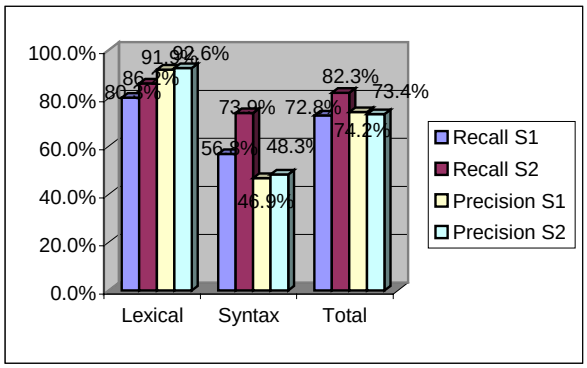

Figure 1. Average recall and precision for detection, system 1 (S1) and system 2 (S2).

We described our user to be specially annoyed by false alarms; the score obtained for precision is thus significant, if the figure is below 75% we estimate that the system is not accurate enough.

On average, precision for S1 amounts to 74.2% while S2 obtains 73.4%. The difference between the lexical and the syntax average are quite significantly high, specially for S1. This points out again, that the average figure, is not enough to evaluate a system, as the two systems appear quite equal on a general average level, while as the detailed results, indeed point out that S1 performs better.

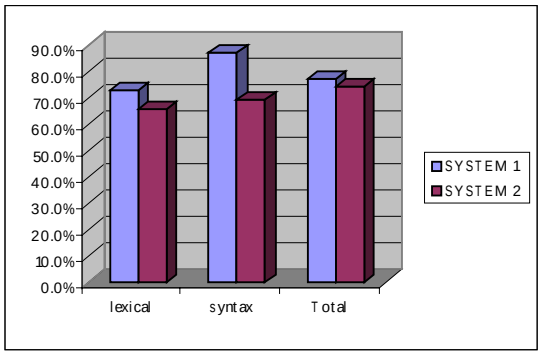

Figure 2. Average percentage of good suggestions for S1 and S2

The same comment could be made about the average percentages for correction. Regarding only the total, which some up the percentages of all errors, the difference between the systems does not appear clearly, although again, the detailed scores are clearly higher for S1.

5. Conclusion and Further Work

The point of the evaluation methodology is to enable a translator to be able himself to determine which system would suit him best. The average scores obtained by the systems are quite close to each other, so that it is not possible to decide on one system or the other on this base. The user would have to determine according to the type of mistakes he is more likely to commit.

The scores given on sentence level gave the accurate impression of S1 performing better. But the other measures are absolutely necessary in order to have a more nuanced and accurate image of the systems performance.

The methodology could be improved through the following changes:

(1) automating the counting phase would increase objectivity and certainly accuracy of the results. This would also make it possible to treat a larger corpus in a reasonable time.

(2) enlarging the corpus or maybe including some test suites in order to study more profoundly certain observations made from the corpus.

(3) unfortunately automating the annotation seems difficult, but if possible, a certain flexibility in the tag setting should be provided to the user, as the systems do not treat or present the corrections in a uniform or even analogous fashion.

This study was limited on French SGCs systems. It would be interesting to gather a similar corpus in different languages, so that the method could be applied to other

languages. Of course this also implies adjusting the error typology to the language treated. In any case, the metrics described in this paper can be used as they are, independently of the language treated.

6. References

Eagles evaluation work group, 1996. *EAGLES Evaluation of Natural Language Processing Systems, Final report*, Center for Sprogteknologi, Denmark.

ISO/IEC 9126-1, 2001. *ISO/IEC 9126-1:2001 (E) Standard – Software engineering - Product quality - Part 1: Quality model*, Geneva: International Organization for Standardization / International Electrotechnical Commission.

ISO/IEC 14598-5, 1998. *ISO/IEC 14598-5:1998 (E) Standard – Information technology --- Software product evaluation --- Part 5: Process for evaluators*, Geneva: International Organization for Standardization / International Electrotechnical Commission.

Kukich, K., 1992. "Techniques of Automatically Correcting Words in Text", in *ACM Computing Surveys*, 24(4):377-438.

Starlander, M. 2002. *Evaluation d'un correcteur d'orthographe et de grammaire : L'utilité de Cordial 7 pour le traducteur*. ETI, Université de Genève, janvier 2002.

TEMAA, 1996. *TEMAA Final Report - A Testbed Study of Evaluation Methodologies: Authoring Aids*. Technical Report LRE-62-070 (March 1996), Center for Sprogteknologi, Copenhagen, Denmark.

Vandeventer, A., Granger, S., Hamel, M.-J., 2002. *Analyse du corpus d'apprenants pour ELAO basé sur le TAL*. In Revue TAL, Hermès, Paris, 42(2).