Baltic J. Modern Computing, Vol. 4 (2016), No. 2, 203-217

Detecting Grammatical Errors in Machine Translation Output Using Dependency Parsing and Treebank Querying

Arda TEZCAN, Véronique HOSTE, Lieve MACKEN

Department of Translation, Interpreting and Communication,
Ghent University, Groot-Brittanniëlaan 45, 9000 Ghent, Belgium

{arda.tezcan, veronique.hoste, lieve.macken}@ugent.be

Abstract. Despite the recent advances in the field of machine translation (MT), MT systems cannot guarantee that the sentences they produce will be fluent and coherent in both syntax and semantics. Detecting and highlighting errors in machine-translated sentences can help post-editors to focus on the erroneous fragments that need to be corrected. This paper presents two methods for detecting grammatical errors in Dutch machine-translated text, using dependency parsing and treebank querying. We test our approach on the output of a statistical and a rule-based MT system for English-Dutch and evaluate the performance on sentence and word-level. The results show that our method can be used to detect grammatical errors with high accuracy on sentence-level in both types of MT output.

Keywords: Machine Translation, Quality Estimation, Dependency Parsing, Treebanks

1. Introduction

Despite the continuous progress that has been made over the last decades, Machine Translation (MT) systems are far from perfect. Moreover, MT quality not only varies between different domains or language pairs but also from sentence to sentence. Even though it has been shown that using MT leads to productivity gains in computer-assisted translation (CAT) workflows (Guerberof, 2009; Depraetere et al., 2014), to produce high-quality translations, humans still need to intervene in the translation process and do this usually by post-editing (correcting) the MT output. Post-editing MT output requires post-editors to detect translation errors prior to correcting them. Hence, automatic quality estimation (QE) systems not only aim to estimate the post-editing effort at segment level to filter low quality translations (Specia et al, 2009), but also to detect the location and the nature of errors at word level (Ueffing and Ney, 2007; Bach et al., 2011).

MT errors can be analysed as adequacy and fluency errors. While adequacy is concerned with how much of the source content and meaning is also expressed in the target text, fluency is concerned with to what extent the translation is well formed and adheres to the norms of the target language. The distinction between adequacy and

fluency has been used in different translation error taxonomies (Lommel et al., 2014; Daems, Macken and Vandepitte, 2014). Besides the difficulties of transferring source content and meaning to a target sentence, the task of producing grammatically correct sentences remains to be challenging for MT systems, independent of the domain of text to be translated and the type of MT system (Costa et al., 2015; Daems et al., 2015). This motivates us to examine the use of dependency structures, which represent the abstract grammatical relations that hold between constituents, for detecting grammatical errors in machine-translated text.

A dependency tree is a rooted, directed acyclic graph, which represents all words in a sentence as nodes and grammatical relations between the words as edges. A labelled dependency tree, additionally, incorporates the nature of the grammatical relationships between the words as annotations of relation names on the edges of the tree. Dependency trees are interesting for the QE task due to the fact that the dependents may span discontinuous parts of the input sentence and are suited for representing languages with word order variations and discontinuous constituencies such as Dutch.

In this paper, we present two new approaches for QE on sub-segment level, and more specifically for detecting grammatical errors in Dutch MT output. In the first approach, we use the partial dependency parses as an indicator of problematic text fragments when no parse covers the complete input. In the second approach, we query the sub-trees of the dependency tree of an MT sentence against a treebank of correct Dutch sentences by using dependency relation and syntactic category information on phrase and lexical level. The number of matching constructions is then considered to be an indicator of possible translation errors for a given sub-tree. In addition to using these two approaches separately, we combine them together and evaluate the three approaches on the output of an English-to-Dutch statistical and rule-based MT system. We evaluate the performance on sentence and word-level by comparing the detected errors to manually annotated errors.

The remainder of this work is as follows: In Section 2, we describe related research. In Section 3, our approach is outlined in detail and in Section 4, we describe the data sets we used. In Section 5, we give the results of our experiments. Finally, in Section 6, we conclude by discussing the results and future work.

2. Related work

QE is the task of providing a quality indicator for machine-translated text without relying on reference translations (Gandrabur and Foster, 2003). Most work on QE has focused on segment level, which aims to provide a binary or continuous quality score for the whole machine-translated sentence that reflects the post-editing time or the number of edits that are required to correct the MT output (Blatz et al., 2004; Specia et al., 2009; Hardmeier et al., 2011). QE on word or sub-segment level, on the other hand, has received less attention.

Estimating the quality of MT output on word or sub-segment level has a number of advantages compared to sentence-level QE. First of all, word-level QE systems can highlight problematic text fragments in machine-translated text to guide the post-editors. Furthermore, since the overall quality of an MT system depends on the individual errors it makes, word-level QE systems can easily be extended to estimate segment-level quality (de Souza et al., 2014; Tezcan et al., 2015). Word-level QE systems can additionally be used for improving MT quality by providing valuable information about

the location, the frequency and the type of errors MT systems make (Popovic and Ney, 2011) or by combining correct text fragments from different MT systems (Ueffing and Ney, 2007).

In one of the early works on word-level QE, Blatz et al. (2004) used a collection of features to build a binary classifier that provides confidence scores for each word in machine-translated text. Besides using features that capture the relationships between source and target words, such as the word posterior probabilities and semantic similarities, they used additional target-language features that were based on basic syntax checking and word frequency. A recent work on word-level QE is QuEst++ (Specia et al., 2015), which is an open-source toolkit for QE on word, sentence and document level. QuEst++ consists of two main modules: a feature extraction and a Machine Learning (ML) module. For word-level QE, besides using features that explore the word alignments and POS-similarities between source and target texts, QuEst++ uses target-language features that are derived from n-gram language models. As n-gram language models rely primarily on local context, they can capture short-distance dependencies (e.g. article-noun agreements), but they fail to capture long-distance dependencies such as non-adjacent subject-verb agreements. The idea of using dependency structures in QE is not new. Bach et al. (2011) compared the dependency relations of Arabic source sentences and English MT translations and incorporated child-father and children correspondences as features in their ML system. They used source and target dependency structure features together with source-side and alignment content features to train a classifier for predicting word-level quality. Hardmeier et al. (2011) used tree kernels over constituency and dependency parses of MT input and output in conjunction with Support Vector Machine (SVM) classification for QE.

A number of studies focused on detecting grammatical errors. Stymne and Ahrenberg (2010) used a mainly rule-based Swedish grammar checker not only to assess the grammaticality of their English-Swedish SMT system, but also for post-processing the MT output by applying the grammar checker suggestions. Ma and McKeown (2011), on the other hand, used feature-based lexicalized tree adjoining grammars (FB-LTAG) to detect and filter ungrammatical translations generated by their MT system. A Tree Adjoining Grammar (TAG) consists of a number of elementary trees, which can be combined with substitution and adjunction operations and while the derivation trees in TAG resemble dependency structures, the derived trees are phrase-structure trees (Joshi and Rambow, 2013).

The approaches we propose differ from previous work in several ways. First of all, we use only dependency tree information of the target language to detect grammatical errors in the MT output and by doing so we make a clear distinction on the type of MT errors we target. Second, in this exploratory study, we do not consider the QE task as a ML problem. Instead we try to gain insights in the strengths and weaknesses of the information the dependency structures provide for the QE task on sub-segment level, so that we can incorporate informative features in a ML system in the future. From this perspective, this method shows similarities to the GrETEL tool described by Augustinus et al. (2012), which allows users to query Dutch strings against a treebank and search for similar syntactic constructions. This application uses XPath[1] for treebank querying and

[1] http://www.w3.org/TR/xpath

can extract sub-trees from full parse trees to allow partial matching. And finally, we evaluate the error detection system against a corpus of MT errors (containing fine-grained manual error annotations), which allows us to focus on grammatical errors only. The evaluation method we use therefore does not involve post-editing speed or number of edits that post-editors make as quality indicators, both of which can be subject to noise due to the changes made in the MT output that are not related to errors.

3. Detecting grammatical errors in machine-translated text

The two error detection approaches that we propose make use of the Alpino parser (van Noord, 2006), a wide-coverage Head-driven Phrase Structure Grammar (HPSG) for Dutch. Alpino constructs for each input sentence a parse forest containing all possible parses. If Alpino is unable to build a parse covering the complete input, the best sequence of non-overlapping parses (each spanning a maximal portion of the input) from this forest is selected (van Noord, 2001). In the first approach, this best sequence of non-overlapping partial parses is used for error. In the second approach, the sub-trees of the final parse tree are extracted and queried against a treebank that contains dependency parses of domain-specific correct Dutch sentences.

3.1. Partial parses

Under the assumption that the parser is accurate enough (Van Noord (2006) reports F-scores of 88.5 or higher, evaluated on different test sets), the fact that Alpino cannot generate a parse for the complete sentence might be an indication of grammatical errors. In the first approach, we simply consider the boundaries of the partial parses as an indicator of errors in the MT output and mark the first *n* words to the left and right of the parse boundaries as errors. This approach uses Alpino-specific output and can only be adapted to other parsers if they output partial parsing information. While choosing different *n* values do not have an impact on the error detection performance on sentence-level, it has an impact on the word-level evaluation, as the higher *n* values means a higher number of words being annotated in the MT output. We discuss the impact of choosing different *n* values on error detection performance in Section 5. Figure 1 shows (a) an English sentence and (b) the Alpino parse of the corresponding MT output in Dutch, in which the boundaries of the partial parses are indicated by means of square brackets. With *n* set to 1, the words in bold are considered as erroneous words.

(a) Unfortunately, these women rarely have the financial means to pay for it.
*(b) [**Helaas**][,][**deze** vrouwen hebben zelden de financiële middelen om te **betalen**][**voor**][**het**][.]*

Figure 1: (a) An English sentence, (b) Alpino parse of the corresponding MT output in Dutch. Square brackets indicate the non-overlapping partial parses and the erroneous words are highlighted in bold (with *n* = 1). Correct Dutch sentence: “Helaas hebben deze vrouwen zelden de financiële middelen om ervoor te betalen”.

3.2. Treebank querying

The Alpino parser constructs a single XML tree and provides categorical information at the level of syntactic constituency and dependency information containing the semantic relations between constituents (Schuurman, 2003). The Alpino XML additionally contains detailed Part-of-Speech (POS) information at the lexical level using the CGN/D-COI tagset (Van Eynde, 2005). An example dependency tree obtained from Alpino is shown in Figure 2 and the Alpino XML for the highlighted sub-tree in Figure 3, which ignores some attributes for expository purposes.

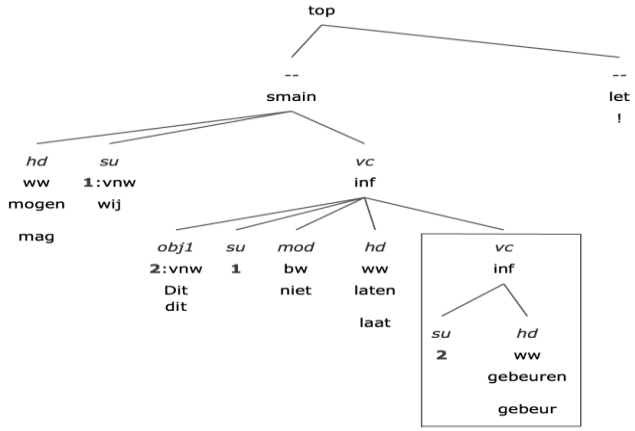


Figure 2. The Alpino dependency tree for the sentence "Dit mogen wij niet laten gebeuren! (En: *We cannot let this happen!*)"

```
<node begin="0" cat="inf" end="6" rel="vc">
    <node begin="0" end="1" index="2" rel="su"/>
    <node begin="5" end="6" lemma="gebeuren" postag="WW(inf,vrij,zonder)" rel="hd"
                word="gebeuren"/>
</node>
```

Figure 3. The Alpino-XML structure for the sub-tree highlighted in Figure 2

Some of the important attributes in the Alpino-XML are *cat* (syntactic category), *rel* (grammatical function), *postag* (part-of-speech tag), *word*, *lemma*, *index*, *begin* (starting position) and *end* (end position). The *index* attribute is used to encode control relations by means of co-indexing. The *postag* attribute in the Alpino-XML additionally provides sub-features for a given lexical item. For example, in Figure 3, the *postag* for the word "gebeuren (*happen*)" is given as "*WW(inf,vrij,zonder)*", which categorizes the word as a verb (*WW*) and more specifically as an infinitive (*inf*), position as free (*vrij*) and inflection as none (*zonder*). To detect errors in the MT output, we collect different sub-trees of depth 1 for a given parse tree and query these against the treebank to search for similar constructions. The sub-trees do not contain any explicit information about the surface forms of words and consist of the following types of information:

- the *rel*, *begin* and *end* attributes for each node

- the *postag* attribute for terminal nodes and *cat* attribute for non-terminal nodes, when available[2]

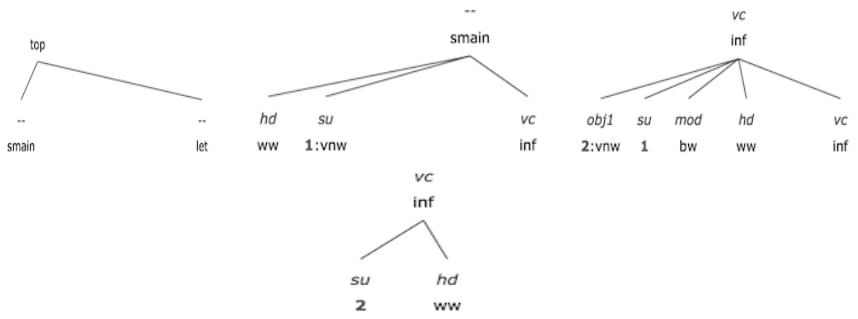
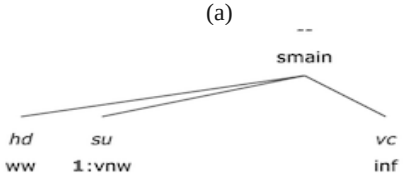


Figure 4. The collected sub-trees from the full parse tree of Figure 2.

(a)



(b)

```
//node[@rel="--" and @cat="smain" and node[@postag="WW(pv,tgw,mv)" and @rel="hd" and
node[@postag="VNW(pers,pron,nomin,vol,1,mv)" and @rel="su" and node[@rel="vc" and @cat="inf"]
```

(c)

```
//node[@rel="--" and @cat="smain" and node[@postag="WW(pv,tgw,mv)" and @rel="hd" and @begin >
../../node[@rel="--" and @cat="smain"]/@begin and @end =
../node[@postag="VNW(pers,pron,nomin,vol,1,mv)" and @rel="su"]/@begin] and
node[@postag="VNW(pers,pron,nomin,vol,1,mv)" and @rel="su" and @begin =
../node[@postag="WW(pv,tgw,mv)" and @rel="hd"]/@end and @end < ../../node[@rel="--" and
@cat="smain"]/@end] and node[@rel="vc" and @cat="inf" and @begin = ../../node[@rel="--" and
@cat="smain"]/@begin and @end = ../../node[@rel="--" and @cat="smain"]/@end]]
```

Figure 5. An example sub-tree (a) and the corresponding XPath query without (b) and with (c) word order constraints indicated by *@begin* and *@end* arguments

Next, for each collected sub-tree a query in XPath is generated. The XPath[3] standard implements a query language for XML documents, which can be used for searching and extracting information from treebanks, using applications like BaseX. The generated XPath query for each node is additionally enriched with starting and end positions to respect the exact word order observed in a given MT output, using the begin and end attributes of the XML. This extension is done to be able to query the treebank not only

[2] Nodes consisting of control relations do not explicitly contain *postag* or *cat* attributes
[3] http://basex.org

for similar constructions with respect to the dependency relation, syntactic category and POS, but also for the order of the constituents. Figure 5 shows (a) the second sub-tree in Figure 4, the corresponding basic XPath query (b) and the extended XPath query that respects the word order of the given structure (c).

Once the XPath queries are generated for each sub-tree, they are used to query the treebank to search for similar constructions and mark errors using BaseX. The motivation for treebank querying is that if a given sub-tree, which consists of dependency relation, syntactic category, POS and word order information, has occurred in large corpus of dependency trees less than a certain threshold value *T*, the sub-tree is grammatically incorrect. In that case we mark all words of the sub-tree as erroneous. We discuss the impact of choosing different *T* values on error detection performance in Section 5. This method is applicable to other parsers (such as the Stanford Parser (De Marneffe et al., 2006)) provided that the output can be converted to the Alpino XML structure or a similar XML structure.

4. Data sets

We use two types of data sets: one data set to evaluate the error detection systems and one data set to construct the treebank. We use the SCATE taxonomy and corpus of MT errors (Tezcan et al., in press) to evaluate the performance of the different error detection approaches. The SCATE error taxonomy is a hierarchical taxonomy, which distinguishes between adequacy and fluency error annotations: errors that can be detected on the target sentence alone (on monolingual level) are considered as fluency errors, whereas errors that can only be detected by looking at both source and target sentences are considered as accuracy errors. Based on the hierarchical structure, both accuracy and fluency errors are categorized further. The fluency errors, being the focus of this study, are further divided into *grammar*, *lexicon*, *orthography*, *multiple errors* and *other fluency errors*. While the first three sub-categories are based on common linguistic notions, *multiple errors* correspond to a combination of fluency errors that are difficult to identify separately, e.g. a word order error combined with wrong word forms or wrong lexical choices. *Other fluency errors* refer to errors that do not belong to any other fluency error categories. Multiple annotations on the same text span are possible.

The SCATE corpus of MT errors is a collection of sentences from the Dutch Parallel Corpus (Macken et al., 2011) consisting of 698 source sentences in English and the MT output for each source segment in Dutch, obtained from two types of MT architectures, namely a Statistical Machine Translation (SMT) System (Google Translate[4]) and a Rule-Based Machine Translation (RBMT) System (Systran[5]). The sentences in this corpus belong to three text types: external communication, non-fictional literature and journalistic texts. Table 1 gives an overview of the number of segments containing errors and the number of error annotations for each data set in the SCATE corpus of MT errors. Based on the definitions of the SCATE fluency error categories, we used all sentences (698) in the SCATE corpus of MT errors but kept only the *grammar* and *multiple error* annotations to assess the performance of the proposed approaches on detecting grammatical errors.

[4] http://translate.google.com
[5] SYSTRAN Enterprise Edition, version 7.5

Table 1. Number of segments containing errors (#segments) and the number of error annotations (#annotations) in the SCATE corpus of MT errors, per error category, per data set. The same information is additionally provided for the merged annotation set of *grammar* and *multiple errors*.

	SMT		RBMT	
	#segments	**#annotations**	**#segments**	**#annotations**
Fluency errors (total)	**536**	**1524**	**563**	**1806**
Grammar errors	435	936	421	855
Orthography errors	192	243	223	284
Lexicon errors	176	232	329	527
Multiple errors	106	112	123	140
Other fluency errors	1	1	0	0
Grammar + Multiple Errors	**463**	**1048**	**451**	**995**

To build the treebank, we used 160,201 Dutch sentences from the Dutch Parallel Corpus. These sentences were collected from the same three text types that are used in the SCATE corpus of MT errors but do not include the sentences that are subject to evaluation. All sentences were automatically parsed with the Alpino parser. An XML database was created from this collection of parse trees using BaseX, in which the XML attributes are indexed. BaseX was also used to make XPath queries against this database of parse trees to mark errors.

5. Experiments and results

Using the error detection approaches described in Section 3, we built three different MT error detection systems: a system which uses the partial parses obtained from the Alpino parser (*P*), a system which uses the matches obtained from the treebank for each sub-tree being queried against (*X*) and a system which combines the output from the first two systems (*P+X*). We evaluate the output of these three systems both on sentence and word level. The sentence-level evaluation is used to assess whether the QE systems can detect sentences containing errors, whereas the word-level evaluation is used to assess whether the systems can locate the errors in the sentence.

5.1. Sentence-level evaluation: Grammar Errors

As the sub-tree extraction method we propose in Section 3.2 does not impose any constraints on the maximum number of child nodes a sub-tree can contain, the *X* system

Table 2. Number and the percentage of sub-trees in the treebank, with a specific number of child nodes.

	N=1	N=2	N=3	N=4	N=5	N=6	N>=7
Number of subtrees	287	156426	847204	452830	145873	49287	18981
Percentage of subtrees	0%	9%	51%	27%	9%	3%	1%

is subject to data sparsity especially if the queried sub-trees contain a high number of child nodes. This problem is clearly visible in the distribution of sub-trees with different number of child nodes (*N*) over all the trees in the treebank, in Table 2.

As it can be seen from Table 2, 96% of all the sub-trees that occur in the treebank contain five of less child nodes. We can therefore expect the *X* system to erroneously flag errors in sub-trees consisting of a higher number of child nodes even though they do not contain grammatical errors, but due to the fact that such sub-trees never occur in the treebank. The first evaluation we make therefore aims to measure the error detection performance of different versions of the X system that query only the sub-trees that consist of equal or less child nodes than the given threshold *MAXN*. The precision, recall and F1 scores for each *X* system are provided in Figure 6, for the SMT and the RBMT output.

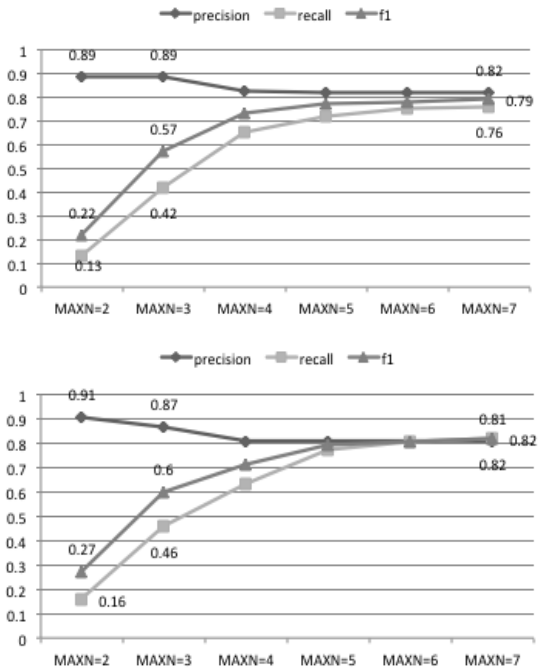


Figure 6. Sentence-level evaluation of the different versions of the system X, with respect to MAXN on the SMT (above) and the RBMT (below) output: Precision, Recall and F1 scores for detecting grammar errors

As MAXN values can be combined with different threshold values *T*, various versions of the *X* system can be built. To have a better understanding of the impact of different *T* values on the error detection performance, we choose an *X* system with high precision (MAXN=3) on both types of MT output and refer to this as the *X* system in the remainder of this study. A simple analysis of the annotations obtained by this system shows us that 10% and 11% of the annotations (SMT and RBMT respectively) marked non-adjacent words and these annotations were able to capture agreement errors such as

the determiner-noun agreement problems as in "onze medisch project *(EN: our medical project)*", which should be rephrased as "ons medisch project".

Next, we evaluate the performance of the three types of error detection systems (*P*, *X* and *P+X*) on detecting grammatical errors at the sentence level, and compare the results for the SMT and the RBMT output in Figures 7 and 8. For the *X* system, we include the results obtained from the different versions, using different *T* values as threshold.

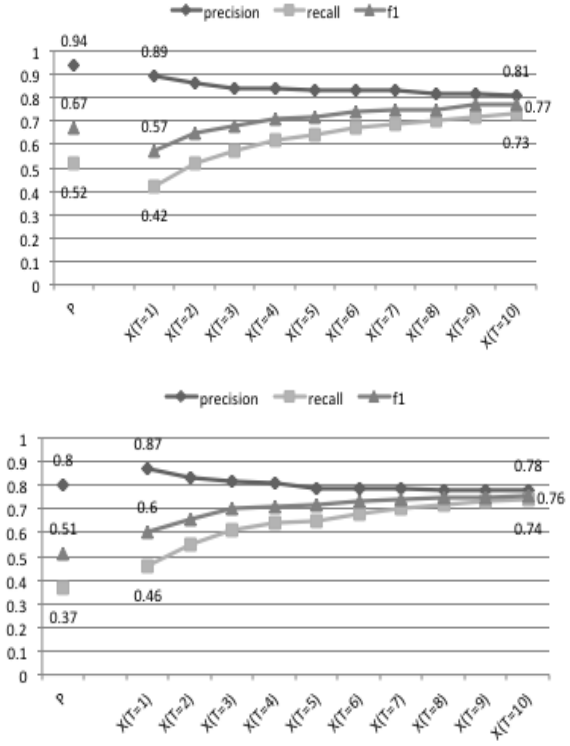


Figure 7. Sentence-level evaluation of the two systems *P* and *X* on SMT (above) and RBMT (below) output: Precision, Recall and F1 scores for error detection performance on grammar errors

We can make a number of observations from Figure 7. First of all, the *P* system performs better overall on detecting grammatical errors in the SMT output compared to the RBMT output. While the *X(T=1)* system shows a higher precision for the SMT system, it shows higher recall for the RBMT system. When we evaluate the *X* system using increasing *T* values, we see a similar trend for both types of MT output, namely minor losses in precision and major gains on recall, which leads to increased F1 scores. However, with increasing *T* values, the system potentially annotates more words per sentence, which should be taken into account for error detection on word-level. We discuss the impact of high *T* values on the performance of word-level error detection in Section 5.2.

If we compare Figures 7 and 8, we can see that the third error detection system (*P+X*) has a higher recall for both types of MT output and for all values of *T* (for the *X* system), which is an indication that the two types of error detection systems detect different grammar errors. In Figure 8, we see that for both types of MT output, the increasing *T* values brings minor losses on precision and major gains on recall, similar to

our observations from Figure 7. A final comparison on sentence level performance can be made with a trivial baseline system, which would mark all sentences (698) as erroneous. Based on the data statistics provided in Table 1, this baseline would score 0,66 and 0,63 on precision with respect to the evaluations made on the SMT and the RBMT output and score 1 on recall for both systems, yielding F1 scores of 79,5 on the SMT output and 77 on the RBMT output. Even though these results would be comparable to the F1 scores we observe in Figures 7 and 8, the strength of the error detection systems being evaluated in this study is the high precision scores they achieve.

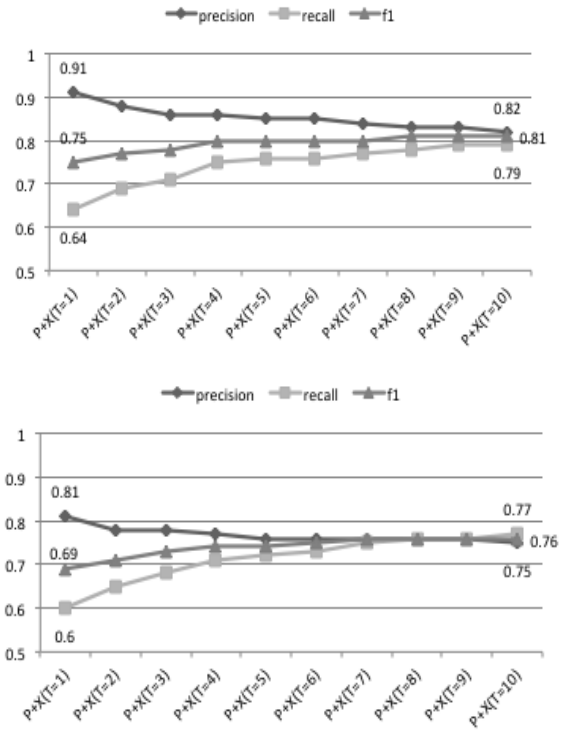


Figure 8. Sentence-level evaluation of the system *P+X* on SMT (above) and RBMT (below) output: Precision, Recall and F1 scores for error detection performance on grammar errors

5.2. Word-level evaluation

We additionally evaluate the two error detection systems *P* and *X* on word level. In this evaluation, each word is considered either erroneous or not and this binary distinction is used to evaluate the error detection performances of the two systems. Since choosing different values of *n* has an impact on the number of errors being marked by the *P* system, in this evaluation, we include different versions of this system, which are based on different *n* values. The evaluation results for detecting errors on word-level are provided in Figure 9, per system.

In Figure 9, we see relatively low precision, recall and F1 scores for all systems. Even though it is difficult to compare, given that we only target grammar errors in this

study, similar F1 results have been observed in the WMT 2015 word-level QE task (Bojar et al., 2015) ranging between 16 and 43. It seems that even though these systems perform well on sentence-level error detection, they are not able to locate the errors within the MT output with high accuracy. So, word-level QE seems to be a more challenging task. One reason for the poorer performance of these systems on word-level error detection can be due to the parsing issues that surface in other parts of the MT output and not on the sub-trees which contain the erroneous words themselves.

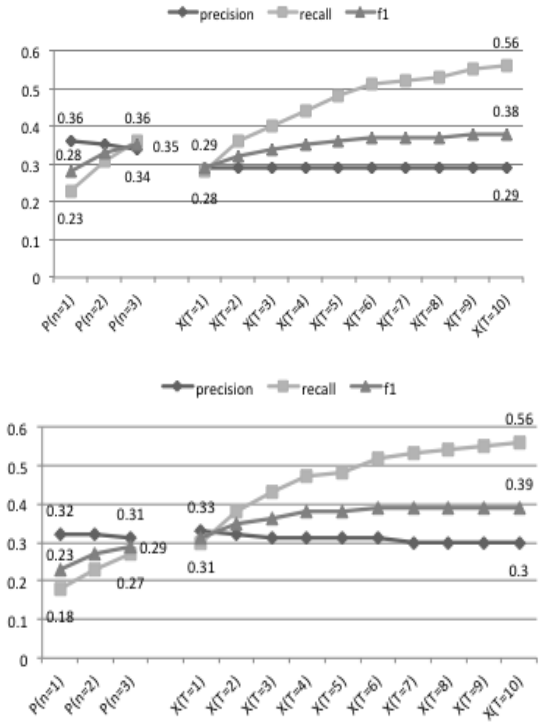


Figure 9. Word-level evaluation of the two systems *P* and *X* on SMT (left) and RBMT (right) output: Performance with respect to precision, recall and F1 scores for locating grammar errors, per system.

Table 3. The average ratio of the number of errors marked erroneous by the error detection systems to the number of words marked in the reference error annotations, per error detection system, per MT type. This comparison is made on the sentences for which there is at least one word being marked as an error both by the error detection systems and in the reference error annotation set.

	P(n=1)	P(n=2)	P(n=3)	X(T=1)	X(T=2)	X(T=3)	X(T=4)	X(T=5)	X(T=6)	X(T=7)	X(T=8)	X(T=9)	X(T=10)
SMT	0,94	1,28	1,58	1,66	1,7	1,73	1,77	1,86	1,88	1,94	1,96	1,97	2
RBMT	0,86	1,14	1,37	1,38	1,57	1,65	1,73	1,76	1,82	1,88	1,89	1,9	1,91

Figure 9 additionally shows us that the *P* system performs better on SMT output (for all performance measures and for all values of *n*). We can also see that the performance of both systems improves, with respect to recall and F1 measures, with increasing *n* and *T* values. However, this increase affects the number of words being marked as error as well as the precision and recall scores. In Table 3, we compare the number of words marked as erroneous between the output of the two error detection systems, *P* and *X*, and the reference error annotations in the evaluation set.

From Table 3, we can see that except for the *P(n=1)* system, all versions of both error detection systems mark more words on average than the number of words being annotated as errors in the reference error annotation set. The optimal values of *n* and *T* would therefore depend on the goal of such an error detection system. If the systems are used for highlighting errors in the MT output, higher values of *n* and *T* can highlight too many words. On the other hand, if the goal is to extract features for predicting post-editing speed, lower values of these parameters could restrict the amount of useful information. Depending on the scenario and the optimal values of *n* and *T*, these two approaches can be combined (as *P+X*) in many different ways.

.

6. Conclusion and future work

We proposed two approaches that use dependency structures for detecting grammatical errors in Dutch MT output. One approach uses the partial parses generated by the Alpino parser, while the second approach uses simple corpus statistics of sub-trees occurring in a treebank. We evaluated the error detection performance of systems based on these two approaches at sentence and word level on SMT and RBMT output and showed that such systems can effectively be used to detect grammatical errors for both types of MT architectures. A third system, which combines the two approaches together, yields especially high precision scores on sentence-level error detection.

While the partial parses generated by the Alpino parser can be considered as information that is language and system-specific, querying sub-trees of automatically generated parse trees against a treebank is a language-independent approach. Moreover, by building the treebank from automatically parsed sentences, we show that this approach does not require manual corrections on parse trees in order to be effective and can be used with larger automatically generated treebanks in combination with queries of sub-trees containing a higher number of nodes in the future. The treebank querying approach currently marks all words spanned under a sub-tree, when an error is detected. This often marks too many words as erroneous. We would like to adapt this method to detect specific grammar errors, such as article-noun or adjective-noun agreements, which can locate the erroneous words alone, without spanning the complete sub-trees in which they appear.

Having shown that the proposed methods provide valuable information about grammatical errors in MT output, we would like to analyse the correlation between the output of these error detection systems with post-editing effort, with respect to post-editing speed and the number of edits that are involved in the post-editing task. Besides being used as standalone tools, these error detection systems can provide useful features for building ML systems. We would also like to use the proposed approaches to generate features of an ML system for QE on sub-segment level. The extracted features can be combined with other fluency features, such as the features obtained from n-gram

language models that can model local context and adequacy features that capture meaning-transfer errors. ML systems can additionally be used to optimize the *n* parameter for the *P* system and *T* and *MAXN* parameters for the *X* system.

Acknowledgements

This research has been carried out in the framework of the SCATE project funded by the Flemish government agency IWT.

References

Augustinus, L., Vandeghinste, V., Van Eynde, F. (2012). Example-Based Treebank Querying. In: *Proceedings of the 8th International Conference on Language Resources and Evaluation*, Istanbul, Turkey, 3161–3167.

Bach, N., Huang, F., Al-Onaizan, Y. (2011). Goodness: A method for measuring machine translation confidence. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Portland, Oregon, USA, 211–219.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., Ueffing, N. (2004). Confidence estimation for machine translation. In: *Proceedings of the 20th international conference on Computational Linguistics*, Association for Computational Linguistics, Geneva, Switzerland, 315.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R. (2014). Findings of the 2014 workshop on statistical machine translation. In: Proceedings of the Ninth Workshop on Statistical Machine Translation Association for Computational Linguistics, Baltimore, MD, USA, 12–58.

Costa, Â., Ling, W., Luís, T., Correia, R., Coheur, L. (2015). A linguistically motivated taxonomy for Machine Translation error analysis. In: Machine Translation, 29(2), 127-161.

Daems, J., Macken, L., Vandepitte, S. (2013). Quality as the sum of its parts: A two-step approach for the identification of translation problems and translation quality assessment for HT and MT+ PE. In: Proceedings of MT Summit XIV Workshop on Post-Editing Technology and Practice, 63–71.

Daems, J., Vandepitte, S., Hartsuiker, R., Macken, L. (2015). The impact of machine translation error types on post-editing effort indicators. In: 4th Workshop on Post-Editing Technology and Practice, Association for Machine Translation in the Americas, Miami, Florida, USA, 31–45.

De Marneffe, M. C., MacCartney, B., Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In: Proceedings of the 5th International Conference on Language Resources and Evaluation, Genoa, Italy, 449–454.

de Souza, J. G., Politecnica de Valencia, U., Buck, C., Turchi, M., Negri, M. (2014). FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task. In: Proceedings of the Ninth Workshop on Statistical Machine Translation, Baltimore, USA, 322–328.

Depraetere, I., De Sutter, N., Tezcan, A. (2014). Post-edited quality, post-editing behaviour and human evaluation: a case study. In: Post-editing of machine translation: processes and applications, 78–108.

Guerberof, A. (2009). Productivity and quality in MT post-editing. In: MT Summit XII-Workshop: Beyond Translation Memories: New Tools for Translators MT, Ottawa, Ontario, Canada, 8.

Hardmeier, C. (2011). Improving machine translation quality prediction with syntactic tree kernels. In: Proceedings of the 15th conference of the European Association for Machine Translation, Leuven, Belgium, 233–240.

Joshi, A., Rambow, O. (2003). A formalism for dependency grammar based on tree adjoining grammar. In: Proceedings of the Conference on Meaning-Text Theory, Paris, France, 207–216.

Lommel, A., Uszkoreit, H. Burchardt, A. (2014). Multidimensional Quality Metrics (MQM): A Framework for Declaring and Describing Translation Quality Metrics. In: Revista tradumàtica: traducció i tecnologies de la informació i la comunicació, (12), 455–463.

Ma, W. Y., McKeown, K. (2011). System Combination for Machine Translation Based on Text-to-Text Generation. In: Proceedings of Machine Translation Summit XIII, Xiamen, China, 546–553.

Macken, L., De Clercq, O. Paulussen, H. (2011). Dutch parallel corpus: a balanced copyright-cleared parallel corpus. In: Meta: Translators' Journal, 56(2), 374-390.

Popović, M., Ney, H. (2011). Towards automatic error analysis of machine translation output. In: Computational Linguistics, 37(4), 657-688.

Schuurman, I., Schouppe, M., Hoekstra, H., Van der Wouden, T. (2003). CGN, an annotated corpus of spoken Dutch. In: Proceedings of 4th International Workshop on Language Resources and Evaluation, Lisbon, Portugal, 340-347.

Gandrabur, S., and Foster, G. (2003). Confidence estimation for text prediction. In: Proceedings of the Conference on Natural Language Learning, Edmonton, Canada, 95–102.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., Cristianini, N. (2009). Estimating the sentence-level quality of machine translation systems. In: Proceedings of the 13th Conference of the European Association for Machine Translation, Barcelona, Spain, 28-37.

Specia, L., Paetzold, G., Scarton, C. (2015). Multi-level Translation Quality Prediction with QuEst++. In: 53rd Annual Meeting of the Association for Computational Linguistics and Seventh International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing: System Demonstrations, Beijing, China, 115-120.

Stymne, S., Ahrenberg, L. (2010). Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation, Valetta, Malta, 2175–2181.

Tezcan, A., Hoste, V., Macken. L. (in press). SCATE Taxonomy and Corpus of Machine Translation Errors. In: Trends in e-tools and resources for translators and interpreters. Leiden: Brill Academic Publishers.

Tezcan, A., Hoste, V., Desmet, B., Macken, L. (2015). UGENT-LT3 SCATE system for machine translation quality estimation. In: Proceedings of the Tenth Workshop on Statistical Machine Translation, Lisboa, Portugal, 353–360.

Ueffing, N., Ney, H. (2007). Word-level confidence estimation for machine translation. In: Computational Linguistics, 33(1), 9–40.

Van Eynde, F. (2005). Part Of Speech Tagging En Lemmatisering Van Het D-Coi Corpus. Intermediate, project-internal version, available at http: //odur.let.rug.nl/vannoord/Lassy/POS_manual.pdf

van Noord, G. (2006). At last parsing is now operational. In: TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles, Leuven, Belgium, 20–42.

van Noord, G. (2001). Robust parsing of word graphs. In: Robustness in Language and Speech Technology, Springer Netherlands, 205–238.

Received May 1, 2016, accepted May 11, 2016