Unsupervised Evaluation of Parser Robustness

Johnny Bigert[1], Jonas Sjöbergh[1], Ola Knutsson[1], and Magnus Sahlgren[2]

[1] KTH Nada, 100 44 Stockholm, Sweden
{johnny, jsh, knutsson}@nada.kth.se
[2] SICS, Box 1263, 164 29 Kista, Sweden
mange@sics.se

Abstract. This article describes an automatic evaluation procedure for NLP system robustness under the strain of noisy and ill-formed input. The procedure requires no manual work or annotated resources. It is language and annotation scheme independent and produces reliable estimates on the robustness of NLP systems. The only requirement is an estimate on the NLP system accuracy. The procedure was applied to five parsers and one part-of-speech tagger on Swedish text. To establish the reliability of the procedure, a comparative evaluation involving annotated resources was carried out on the tagger and three of the parsers.

1 Introduction

Automatic parsing of text is a popular field of research. Many of the applications where parsing is used, such as parsing human input to a computer system, handle text that is not proofread. Depending on the application, the text can be relatively error free (e.g. parsing newspaper articles from the internet) or contain large amounts of errors (e.g. using a parser as a tool for second language learners when writing essays). If the intended use of a parser is domains with many errors, it must be robust enough to produce useful output despite noisy input. It is not sufficient to achieve a good performance on error-free text. Usually, the accuracy of a parser on error-free text is known, but the accuracy on texts containing errors is often unknown.

Carroll and others give a comprehensive overview of different parser evaluation methods and discuss some shortcomings [1]. Evaluation of parsers is usually carried out by comparing the parser output to a manually annotated or manually corrected version of a test text. Manual work is expensive, and not necessarily error free. If the NLP system is under development, the evaluation has to be carried out repeatedly. Thus, very large amounts of annotated resources may be required to avoid data exhaustion. Many languages have no large manually annotated resources at all, and those existing often contain only error-free texts.

Manual annotation is not only expensive, but often hard to reuse when evaluating a new parser. Generally, it is non-trivial to map the output of one parser to the output of another [2]. Thus, the effort of manually annotating text with one type of parse information is not generally reusable for other parsers.

To carry out the evaluation of NLP system robustness while avoiding the abovementioned drawbacks, we propose a procedure that requires no manual work or annotated

A. Gelbukh (Ed.): CICLing 2005, LNCS 3406, pp. 142–154, 2005.

resources. There are, as pointed out by Menzel [3], many types of robustness. Robustness in this context is defined as the system's reluctance to change its output when the input becomes increasingly noisy and ill-formed. The only requirements of the evaluation method are a (relatively error-free) text and an estimate of the accuracy of the parser (on error-free text, which is usually known). Despite the modest requirements, the evaluation procedure provides accurate estimates of the robustness of an NLP system.

The method is an extension of a supervised approach to parser robustness evaluation [4]. It is unsupervised and based on introduction of artificial spelling errors in error-free text. We have chosen to use spelling errors to simulate noisy input for several reasons. First, performance (keyboard) spelling errors are language independent. Hence, anyone can use the framework and apply it to their parser in their language without modification. Second, performance spelling errors are easily described and widely understood and thus, does not obscure the important parts of the evaluation procedure. Also, to keep the description of the error model as straightforward as possible, we have refrained from applying an automatic spelling corrector. Please keep in mind that the evaluation method is not restricted to spelling errors, but applicable to any error type, such as incomplete sentences in the sense of e.g. [5].

Another approach to evaluation of parser robustness is provided by Foster [6]. There, parser robustness is evaluated by running a parser on ungrammatical text and comparing the output to the output when run on the same text after it has been manually corrected. Also, Li [7] proposes a method based on an annotated corpus of low quality language use, in this case transcribed phone calls.

We assessed the reliability of the evaluation method by using five different parsers and one part-of-speech tagger. All five parsers process written Swedish text, even though the evaluation method is language independent. The tagger and three of the parsers had resources annotated with the correct tagger/parser output, allowing us to verify the results of the unsupervised evaluation.

2 Proposed Method

We are given an NLP system processing and outputting row-based data, that is, reading one input (e.g. a word) per row and producing one output (e.g. a parse string) per row. We want to assess the robustness of the system. To this end, we need to evaluate the performance of the system when applied to input with increasing amounts of noise. The proposed method is applicable to most NLP system, but parsers will be used here for the clarity of exposition.

Naturally, the performance of an NLP system can be better assessed with an annotated resource. To begin with, the discussion here will include such a resource. The aim is to establish how much information can be gained concerning the performance of the NLP system *without* the annotated resource.

We require a text to be used in the evaluation. The text will be processed by the NLP system (i.e. a parser). Even though the text can be chosen arbitrarily, we simplify the exposition of the method by using the text from a treebank; but keep in mind that the method does not require an annotated resource. We introduce spelling errors in the text to determine the performance of the NLP system under the influence of noisy and ill-formed input. To this end, we use a freeware program called MISSPLEL [8], producing human-like spelling errors. We introduce spelling errors simulating keyboard mistypes. To avoid alternate interpretations of a sentence, the spelling errors result only in words not present in a dictionary. For more details on the introduction of spelling errors, we refer to [4].

Three different data sources are involved in the discussion of the evaluation method. The three files have the same number of rows since they all originate from the same text (i.e. the text in the treebank). For each row, they contain a data pair: a word (that may or may not be misspelled) and a parse string for that word. Only the parse part is used here.

The first file, denoted m, is the manually checked annotated resource (e.g. a tree bank). The second file, denoted 0 (zero), is the output of the NLP system when applied to the original treebank text (0% errors). The third file, denoted n, is the output of the NLP system when applied to the text containing errors (e.g. $\texttt{n} = 5\%$ of the words in the file are misspelled). Clearly, a file containing $\texttt{n}\%$ errors is more difficult to parse than an error-free text and we want to determine how difficult.

2.1 Five Cases

Given one row of the treebank, the 0% file and the $\texttt{n}\%$ file, we analyze the different cases that may occur. Say that the treebank parse (i.e. the correct answer) is a. The 0% file either contains the correct answer a, or an incorrect answer b. Furthermore, the $\texttt{n}\%$ file may contain the correct answer a, the same incorrect answer as the 0% file b or even another incorrect answer c. From this, we obtain several different combinations.

We introduce a notation (denoted m0n) consisting of three columns. The first position is the parse found in the treebank m, the second is the 0% file 0 and the third is the $\texttt{n}\%$ file n. For example, abc means that the parse from the treebank was a, the parse from the 0% file was b and the parse found in the $\texttt{n}\%$ file was c.

Thus, using the new notation, we get five different cases when comparing parses of a single word: aaa, aab, aba, abb and abc. See Table 1 for an example. The first case aaa is the most common, where all three files agree on the same parse. Second, aab is the case where an error nearby in the text corrupted the parsing process of this row. The third case aba is unusual, but not negligibly so. This may occur when the parser is uncertain and chooses between two equal alternatives and arbitrarily chooses the correct one at the $\texttt{n}\%$ level due to a nearby error in the text. The fourth case abb is common and occurs when the parser does not know how to parse a correct grammatical construction. The last case abc may be caused by an error introduced near a correct grammatical construction that the parser cannot parse correctly. This case is uncommon.

Let $x_{aaa}, x_{aab}, x_{aba}, x_{abb}$ and x_{abc} correspond to the relative frequencies of the five cases. For example, if abb occupies 10% of the rows, $x_{abb} = 0.10$. Clearly,

$$x_{aaa} + x_{aab} + x_{aba} + x_{abb} + x_{abc} = 1, \tag{1}$$

since they cover all possible outcomes. Let acr_{m0} denote the accuracy when comparing the m file (treebank) to the 0 file (error-free text). We see that

$$acr_{m0} = x_{aaa} + x_{aab} \tag{2}$$

Table 1. An example of the different cases resulting from parsing a single word. Translation: Vi (We) kan (can) välja (choose) att (to) säga upp (cancel) avtalet (the agreement)

(treebank) word	manual annotation	(error-free text) word	parser output	(n% errors) word	parser output	case
Vi	NP begin	Vi	NP begin	Vi	NP begin	aaa
kan	VP begin	kan	VP begin	*kna*	VP begin	aaa
välja	VP end	välja	VP end	välja	**VP begin**	aab
att	NP(inf) begin	att	**Outside**	att	NP(inf) begin	aba
säga	VP begin in NP	säga	**VP begin**	säga	VP begin in NP	aba
upp	VP end in NP	upp	**VP end**	*upö*	**NP begin in NP**	abc
avtalet	NP begin in NP	avtalet	**NP begin**	avtalet	**NP begin**	abb

since only in cases aaa and aab, the two columns m and 0 contain the same output a. Furthermore, by the same reasoning,

$$acr_{mn} = x_{aaa} + x_{aba} \quad \text{and} \tag{3}$$

$$acr_{0n} = x_{aaa} + x_{abb}. \tag{4}$$

The x_{abb} is included in the last equality since 0 equals n in abb even though they both differ from m. The fact that they differ from the treebank cannot be established without the correct answer m.

We say that the performance of the NLP system *degrades* when the performance decreases with increasing levels of errors in the text. The degradation $degr_n$ is a comparison between the performance at the n% error level and the performance at the 0% error level. Let

$$degr_n = 1 - \frac{acr_{mn}}{acr_{m0}}. \tag{5}$$

Clearly, this is calculable only if you have access to acr_{mn} and acr_{m0}.

Normally, some sort of evaluation has been carried out to estimate the accuracy of the parser on error-free text, denoted *acr*. High accuracy is obtained when the correct answer m often corresponds to the output 0. Thus, the accuracy is a very good estimate for acr_{m0} and we will use $acr_{m0} = acr$. Nevertheless, without the annotated resource, we do not have access to or estimates for acr_{mn}.

2.2 Upper and Lower Bounds

We want to estimate the degradation $degr_n$ without knowing acr_{mn}. Without the annotated resource, we only have access to acr_{0n} and $acr_{m0} = acr$. We will use these to establish an upper bound $degr_n^{upr}$ for $degr_n$. We want the value $degr_n^{upr}$ to be an expression including *acr* and acr_{0n} that can be proven to be greater than $degr_n$.

We propose

$$degr_n^{upr} = \frac{1 - acr_{0n}}{acr} \tag{6}$$

as an upper bound. We prove that $degr_n^{upr}$ is always greater than $degr_n$ by letting

$$degr_n^{upr} = degr_n + \epsilon. \tag{7}$$

Equations (1)–(2) and (4)–(6) give us

$$\epsilon = \frac{2x_{aba} + x_{abc}}{acr}. \tag{8}$$

We see that $\epsilon \geq 0$ since all $x \geq 0$ and thus, $degr_n^{upr} \geq degr_n$ as required.

The smaller the value of ϵ, the better. From the discussion, we see that x_{aba} and x_{abc} are normally quite small, which is promising.

We now turn to a lower bound for $degr_n$. We propose

$$degr_n^{lwr} = \frac{1}{2} degr_n^{upr} = \frac{1 - acr_{0n}}{2acr}. \tag{9}$$

Again, as for the upper bound, the expression must be proven to be less than $degr_n$. To this end, we let

$$degr_n^{lwr} + \delta = degr_n. \tag{10}$$

From Equations (1)–(2), (4)–(5) and (9), we obtain

$$\delta = \frac{x_{aab} - 3x_{aba} - x_{abc}}{2acr}, \tag{11}$$

which is non-negative when $x_{aab} \geq 3x_{aba} + x_{abc}$.

Both cases aab, aba and abc are the result of an introduced spelling error. With no errors, x_{aab}, x_{aba} and x_{abc} are all zero and with increased levels of introduced errors, they will all increase. Hence, x_{aab}, x_{aba} and x_{abc} are positively correlated. Furthermore, it is clear that case aab is much more common than aba and abc since it involves correctly parsed text at the 0% error level. The accuracy *acr* determines the amount of correctly parsed text and thus, with reasonable accuracy, the above inequality holds with a good margin of error. See Appendix A for details on the conditions under which the above inequality holds. Section 3 further support that the inequality holds, since in all experiments the left-hand side is more than twice the right-hand side.

From the above discussion and given the conditions, we have obtained

$$degr_n^{lwr} \leq degr_n \leq degr_n^{upr}. \tag{12}$$

2.3 Estimation of the Degradation

The simple relationship between the upper and lower bounds allows us to deduce some further information. Given an upper bound $degr_n^{upr}$ and a lower bound $degr_n^{lwr}$, we want to estimate the position of the true value $degr_n$. Clearly, $degr_n$ is somewhere in between $degr_n^{lwr}$ and $degr_n^{upr}$ from Equation (12). Let $degr_n^{est}$ be the center of the interval contained by the lower and upper bound, that is,

$$degr_n^{est} = \frac{1}{2}(degr_n^{lwr} + degr_n^{upr}) \tag{13}$$

and let γ be the distance from $degr_n$ to $degr_n^{est}$. Then,

$$degr_n + \gamma = degr_n^{est}. \tag{14}$$

Equations (7), (10) and (13) yield $\gamma = (\epsilon - \delta)/2$. Using Equations (8) and (11) results in the explicit form

$$\gamma = \frac{7x_{aba} + 3x_{abc} - x_{aab}}{4acr}. \tag{15}$$

We see that γ is small if $7x_{aba} + 3x_{abc} \approx x_{aab}$.

As the discussion above about the lower bound illustrated, x_{aab}, x_{aba} and x_{abc} are correlated. See Appendix A for a discussion on the conditions required to make γ small. Though the experiments in Section 3 show that γ is quite small, we make no claims that γ is equally small for all NLP systems. The estimations here are just theoretical indications where the true value of $degr_n$ may reside.

We have indicated that $degr_n^{est}$ is, in theory, close to $degr_n$. By using Equations (6) and (9), we simplify and obtain an explicit formula for the estimated degradation:

$$degr_n^{est} = \frac{3}{4} degr_n^{upr} = \frac{3(1 - acr_{0n})}{4acr}. \tag{16}$$

Hence, without having an annotated resource, we can estimate the robustness (degradation) of the system quite accurately.

2.4 Accuracy

Now that the degradation of the performance has been established, we turn to the accuracy. The definition of $degr_n$ in Equation (5) states that $degr_n = 1 - acr_{mn}/acr$. We are interested in the accuracy of the NLP system on the $\mathtt{n}\%$ file, that is, acr_{mn}. Rearranging the above equation yields

$$acr_{mn} = acr(1 - degr_n). \tag{17}$$

Since $degr_n$ is unknown, we use $degr_n^{upr}$, $degr_n^{lwr}$ and $degr_n^{est}$ to obtain bounds on the accuracy:

$$acr_{mn}^{lwr} = acr(1 - degr_n^{upr}), \tag{18}$$
$$acr_{mn}^{upr} = acr(1 - degr_n^{lwr}), \tag{19}$$
$$acr_{mn}^{est} = acr(1 - degr_n^{est}). \tag{20}$$

The estimation in Equation (20) is not precise, so we let

$$acr_{mn} + \lambda = acr_{mn}^{est}. \tag{21}$$

From Equations (14), (17) and (20), we obtain

$$\lambda = acr \cdot (-\gamma). \tag{22}$$

Thus, if $|\gamma|$ is small, $|\lambda|$ is even smaller, and thus, acr_{mn}^{est} is a good approximation of the accuracy of the NLP system when applied to a file containing $\mathtt{n}\%$ errors.

3 Empirical Results

Five different parsers were used to assess the accuracy of the evaluation method.

GTA [9] is a rule-based shallow parser. It relies on hand-crafted rules of which a few are context-sensitive. The rules are applied to part-of-speech tagged text. GTA identifies constituents and assigns phrase labels but does not build full trees with a top node.

FDG [10], Functional Dependency Grammar, is a commercial dependency parser. It builds a connected tree structure, where every word points at a dominating word. Dependency links are assigned a function label. FDG produces other information too, such as morphological analysis and lemma of words, which is not used here.

A dependency parser by Nivre [11] uses a manually constructed grammar and assigns dependency links between words, working from part-of-speech tagged text. We denoted it the MCD parser (manually constructed dependency).

The Malt parser [12], another dependency parser, is based on the same algorithm as MCD but uses a memory-based classifier trained on a treebank instead of a manually constructed grammar. Unlike MCD, the Malt parser not only assigns dependency links between words but also attaches function labels to these links.

A manually constructed context-free grammar for Swedish was used with an implementation of Earley's parsing algorithm, as described in [13]. We denoted it the Earley parser.

3.1 Parser Robustness Evaluation

In the evaluation, we used 100 000 words from the Stockholm-Umeå Corpus (SUC) [14]. The SUC is a balanced collection of written Swedish, well proofread. The SUC is annotated with part-of-speech information. It does not contain any parse annotation.

The 100 000 word text was parsed using each of the parsers. The parse results from this error-free text (0% errors) constituted the $\mathtt{0}$ file, as defined in the first part of Section 2. Spelling errors (resulting in non-existing words only) were randomly inserted into the text, using a tool that emulates errors produced by a human, as described in Section 2. The parse results from the misspelled text (containing e.g. 5% errors) constituted the $\mathtt{n}$ file, also from Section 2. For the GTA, the MCD and the Malt parser, manually annotated resources were available. The experiments on these are reported in the next section.

To see how the parser behaves with increasing amounts of errors, $\mathtt{n} = 1\%, 2\%, 5\%$, 10% and 20% of all words were randomly misspelled. To reduce the influence of chance, 10 different misspelled files were created for each error level. Using these, we calculated the mean for the degradation, the accuracy and so forth. The variance between different files was low. To simplify the evaluation, a freeware program called AutoEval [8] was used for input and output handling and data processing.

The degradation estimates for a particular file were obtained by calculating acr_{0n}, that is, by comparing how many of the parses in the $\mathtt{0}$ file that corresponded to the parses in the $\mathtt{n}$ file. From acr_{0n} we calculated the upper and lower bounds as well as estimates on the degradation and accuracy, as seen in Section 2.

The results for the five parsers are presented in Tables 2 through 6, which also present the accuracy *acr* on error-free text. The first column reports on the amount of errors in the text. The second is the amount of parse output that differs between the rows of the $\mathtt{0}$ file and the $\mathtt{n}$ file. This value is $1 - acr_{0n}$. The third column presents the degradation of the parser. The first value is the lower bound $degr_n^{lwr}$ and the second is the upper bound $degr_n^{upr}$. The figure in parentheses is the estimated degradation $degr_n^{est}$. The fourth column contains the estimations on the accuracy: lower bound acr_{mn}^{lwr}, upper bound acr_{mn}^{upr} and estimated value acr_{mn}^{est}.

Table 2. Estimated robustness of the GTA parser on 100 000 words. All figures are given in per cent. Estimated accuracy on error-free text was 89%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	1.2	0.7 - 1.3 (1.0)	88 - 88 (88)
2	2.4	1.3 - 2.6 (2.0)	87 - 88 (87)
5	5.7	3.2 - 6.4 (4.8)	83 - 86 (85)
10	11	6.2 - 12 (9.4)	78 - 83 (81)
20	21	12 - 24 (18)	68 - 78 (73)

Table 3. Estimated robustness of the MCD parser on 100 000 words. Estimated accuracy on error-free text was 82%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	0.9	0.5 - 1.1 (0.8)	81 - 82 (82)
2	1.7	1.1 - 2.1 (1.6)	81 - 81 (81)
5	4.3	2.6 - 5.3 (4.0)	78 - 80 (79)
10	8.6	5.2 - 10 (7.8)	74 - 78 (76)
20	17	10 - 20 (15)	66 - 74 (72)

Table 4. Estimated robustness of the Malt parser on 100 000 words. Estimated accuracy on error-free text was 79%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	1.8	1.2 - 2.4 (1.8)	77 - 78 (77)
2	3.7	2.3 - 4.7 (3.5)	75 - 77 (76)
5	8.9	5.7 - 11 (8.5)	70 - 74 (72)
10	17	11 - 22 (16)	61 - 70 (66)
20	31	20 - 39 (29)	48 - 63 (55)

Table 5. Estimated robustness of the Earley parser on 100 000 words. Estimated accuracy on error-free text was 90%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	0.8	0.5 - 0.9 (0.7)	89 - 90 (89)
2	1.7	0.9 - 1.8 (1.4)	88 - 89 (89)
5	4.1	2.3 - 4.5 (3.4)	86 - 88 (87)
10	8.2	4.5 - 9.1 (6.8)	82 - 86 (84)
20	16	9.1 - 18 (14)	74 - 82 (78)

The proposed method evaluates the robustness on one row at the time. For example, if the first column says 5%, we have introduced errors in 5% of the words (with one word per row). Similarly, if we report 11% in the second column (parse differs), then 11% of the parse output (with one parse per row) is different between the two files.

In the experiments, any deviation from the correct parse was considered an error, even if it was "almost" correct (though the evaluation method could just as easily use a more sophisticated analysis). Hence, parsers that provide richer information will generally be

Table 6. Estimated robustness of the FDG parser on 100 000 words. Estimated accuracy on error-free text was 90%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	2.1	1.2 - 2.3 (1.7)	88 - 89 (88)
2	4.2	2.3 - 4.6 (3.5)	86 - 88 (87)
5	10	5.5 - 11 (8.3)	80 - 85 (83)
10	19	11 - 21 (16)	71 - 81 (76)
20	34	19 - 37 (28)	56 - 73 (65)

Table 7. Estimated robustness of the PoS tagger TnT on 100 000 words. All figures are given in per cent. Estimated accuracy on error-free text was 96%

Error level	Output differs	Estimated degradation	Estimated accuracy
1	0.7	0.4 - 0.7 (0.6)	95 - 96 (95)
2	1.4	0.7 - 1.5 (1.1)	95 - 95 (95)
5	3.6	1.9 - 3.7 (2.8)	92 - 94 (93)
10	7.2	3.7 - 7.5 (5.6)	89 - 92 (91)
20	14	7.5 - 15 (11)	82 - 89 (85)

less robust than parsers that return less information, since there are more possibilities for errors.

Parsers base much of their decisions on the part-of-speech information assigned to a word. Since part-of-speech taggers often guess the correct tag for regularly inflected unknown words, the part-of-speech tagger is responsible for a large part of the robustness. In Table 7, the estimated degradation of the part-of-speech (PoS) tagger TnT [15] is shown. TnT was used for all parsers but FDG, which includes its own tagger.

Comparing the output of FDG on different versions of the same text is non-trivial, since the tokenization may be altered by a misspelled word. Here, any tokens without a directly corresponding token in the other text were ignored. All other tokenization difficulties were interpreted to give FDG as many "correct" parses as possible. The 90% accuracy for FDG is our estimation. Malt and MCD are similar in their construction but their results are not really comparable since Malt assigns function labels and MCD does not. On unlabeled output, Malt is more accurate than MCD.

3.2 Evaluating the Evaluation Method

Text with correctly annotated parse output was available for some of the parsers, though only in small amounts. By using these, we wanted to assess the accuracy of the proposed method.

For the GTA parser and the TnT part-of-speech tagger, we had a 14 000 word file of manually corrected parse and tag data. For the MCD parser, we had a 4 000 word file and for Malt we had 10 000 words. We used the text from these files and carried out the same procedure as in the previous subsection, that is, introduced errors and evaluated. We also had the correct answers from the annotated resource. From this, we calculated the real degradation and accuracy.

Table 8. Estimated and actual robustness of the GTA parser on 14 000 words of manually annotated text. All figures are given in per cent. Estimated parser accuracy on error-free text was 89%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	1.2	0.7 - 1.4 (**1.0**)	**0.9**	88 - 88 (**88**)	**88**
2	2.3	1.3 - 2.6 (**1.9**)	**1.8**	87 - 88 (**87**)	**87**
5	5.1	2.9 - 5.7 (**4.3**)	**4.2**	84 - 86 (**85**)	**85**
10	9.9	5.5 - 11 (**8.3**)	**8.1**	79 - 84 (**81**)	**82**
20	19	10 - 21 (**16**)	**16**	70 - 80 (**75**)	**75**

Table 9. Estimated and actual robustness of the MCD parser on 4 000 words of manually annotated text. Estimated parser accuracy on error-free text was 82%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	0.7	0.4 - 0.8 (**0.6**)	**0.6**	82 - 82 (**82**)	**82**
2	1.7	1.0 - 2.0 (**1.5**)	**1.4**	81 - 82 (**81**)	**81**
5	4.0	2.5 - 4.9 (**3.7**)	**3.2**	78 - 80 (**79**)	**80**
10	8.3	5.0 - 10 (**7.6**)	**6.6**	74 - 78 (**76**)	**77**
20	16	9.6 - 19 (**14**)	**13**	67 - 74 (**71**)	**72**

Table 10. Estimated and actual robustness of the Malt parser on 10 000 words of manually annotated text. Estimated parser accuracy on error-free text was 79%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	1.8	1.1 - 2.3 (**1.7**)	**1.3**	77 - 78 (**77**)	**78**
2	3.4	2.2 - 4.3 (**3.2**)	**2.4**	75 - 77 (**76**)	**77**
5	8.7	5.5 - 11 (**8.3**)	**6.1**	70 - 74 (**72**)	**74**
10	16	11 - 21 (**16**)	**12**	62 - 70 (**66**)	**69**
20	30	19 - 38 (**29**)	**23**	48 - 64 (**56**)	**60**

Table 11. Estimated and actual robustness of the TnT part-of-speech tagger on 14 000 words of manually annotated text. Estimated tagger accuracy on error-free text was 96%

Error level	Output differs	Estimated degradation	**Real degr.**	Estimated accuracy	**Real accur.**
1	1.1	0.6 - 1.1 (**0.9**)	**0.9**	95 - 95 (**95**)	**95**
2	1.9	1.0 - 2.0 (**1.5**)	**1.6**	94 - 95 (**94**)	**94**
5	3.9	2.0 - 4.1 (**3.1**)	**3.6**	92 - 94 (**93**)	**92**
10	7.3	3.8 - 7.6 (**5.7**)	**6.7**	88 - 92 (**90**)	**89**
20	14	7.4 - 15 (**11**)	**13**	82 - 89 (**85**)	**83**

The results are provided in Tables 8 through 11. As guaranteed by the proposed method, the real degradation and accuracy are always between the lower and upper bound. We see that the estimated degradation and accuracy are close or equal to the real degradation and accuracy, as indicated in the discussion about γ in Section 2.3 and λ in Section 2.4. Hence, there is strong reason to believe that the estimations on the 100 000 word files in Section 3.1 are also accurate. Furthermore, by using the results from a small annotated resource (if available), we obtain a good estimate on the relation γ between the real and the estimated degradation for the 100 000 file.

We note that rich information is a liability for at least two of the parsers, FDG and Malt. Thus, comparing the robustness figures between two parsers is not entirely fair. Nevertheless, if the objective is reluctancy to change the output when facing unrestricted and noisy text, the figures are accurate.

We also note that the proposed method could easily be adapted to other types of output besides the row-based used here. This might require a small adjustment of the estimations in the theory section.

4 Conclusions

We have presented a method to estimate the robustness of an NLP system. The method provides lower and upper bounds as well as estimates on the actual robustness. The main strength of the evaluation is that neither manual work nor annotated resources are required. The only requirements are an arbitrary (unannotated) text and an estimate of the accuracy of the parser on error-free text. Thus, we have eliminated the need for expensive and time-consuming manual labor.

The proposed method is applicable to any language and most annotation schemes and NLP systems. Even though spelling errors have been used here as an example in the presentation of the method, any error type can be used to simulate noise. Using annotated resources, we have assessed the reliability of the unsupervised evaluation and found that the estimates were quite accurate. We conclude that the proposed method is a reliable and highly timesaving tool for the evaluation of NLP system robustness.

A Conditions

We want to determine the circumstances under which the restriction on δ holds, that is, when

$$\delta = \frac{x_{aab} - 3x_{aba} - x_{abc}}{2\mathit{acr}} \geq 0, \tag{23}$$

as discussed in Section 2.2. Furthermore, we will establish the requirements for γ to be small, i.e. when

$$\gamma = \frac{7x_{aba} + 3x_{abc} - x_{aab}}{4\mathit{acr}} \approx 0. \tag{24}$$

A few assumptions are required. We know from Equations (1) and (4) that

$$x_{aab} + x_{aba} + x_{abc} = 1 - \mathit{acr}_{0n}. \tag{25}$$

We are interested in an approximation of x_{aab}. We will assume that $x_{aab}/(1 - \mathit{acr}_{0n}) = \mathit{acr}$. That is, we assume that x_{aab} compared to the three cases $x_{aab} + x_{aba} + x_{abc}$ is about the same as the accuracy *acr* compared to one (the sum of all cases). The reader should take a moment to recognize that this is not an unreasonable estimation. We rearrange the above approximation and obtain

$$x_{aab} = \mathit{acr}(1 - \mathit{acr}_{0n}). \tag{26}$$

From this and Equation (25), we get

$$x_{aba} + x_{abc} = (1 - acr)(1 - acr_{0n}). \tag{27}$$

Our second assumption is that

$$x_{aba} \leq x_{abc}. \tag{28}$$

The two cases `aba` and `abc` originate from a grammatical construct that could not be parsed by the system. When an error is introduced, the parser changes its output. The most probable is that the change results in something erroneous, as in `abc`.

We use the assumptions with δ in Equation (23):

$$\begin{gathered}\delta = (x_{aab} - 3x_{aba} - x_{abc})/2acr \geq \\ (x_{aab} - 2(x_{aba} + x_{abc}))/2acr \geq 0 \\ \iff acr - 2(1 - acr) \geq 0.\end{gathered}$$

Hence, the inequality in Equation (23) is satisfied if $acr \geq 2/3$. If we have an accuracy of more than 67%, the lower bound for the degradation is valid.

We repeat the above process with γ in Equation (24) and obtain

$$\begin{gathered}\gamma = (7x_{aba} + 3x_{abc} - x_{aab})/4acr \leq \\ (5(x_{aba} + x_{abc}) - x_{aab})/4acr \leq 0 \\ \iff 5(1 - acr) - acr \leq 0.\end{gathered}$$

Hence, γ in Equation (24) is negative if $acr \geq 5/6 = 83.3\%$. On the other hand,

$$\begin{gathered}\gamma = (7x_{aba} + 3x_{abc} - x_{aab})/4acr \geq \\ (3(x_{aba} + x_{abc}) - x_{aab})/4acr \geq 0 \\ \iff 3(1 - acr) - acr \geq 0.\end{gathered}$$

Now, γ is positive if $acr \leq 3/4 = 75\%$. Thus, for parsers with reasonable accuracy, γ will be small and the approximation of the degradation will be accurate.

References

1. Carroll, J., Briscoe, T., Sanfilippo, A.: Parser evaluation: a survey and a new proposal. In: Proceedings of LREC 1998, Granada, Spain (1998) 447–454
2. Hogenhout, W.I., Matsumoto, Y.: Towards a more careful evaluation of broad coverage parsing systems. In: Proceedings of Coling 1996, San Francisco, USA (1996) 562–567
3. Menzel, W.: Robust processing of natural language. In: Proceedings of 19th Annual German Conference on Artificial Intelligence, Berlin, Germany (1995) 19–34
4. Bigert, J., Knutsson, O., Sjöbergh, J.: Automatic evaluation of robustness and degradation in tagging and parsing. In: Proceedings of RANLP 2003, Bovorets, Bulgaria (2003)
5. Vilares, M., Darriba, V.M., Vilares, J., Rodriguez, R.: Robust parsing using dynamic programming. Lecture Notes in Computer Science **2759** (2003) 258–267
6. Foster, J.: Parsing ungrammatical input: An evaluation procedure. In: Proceedings of LREC 2004, Lisbon, Portugal (2004) 2039–2042
7. Li, X., Roth, D.: Exploring evidence for shallow parsing. In Daelemans, W., Zajac, R., eds.: Proceedings of CoNLL 2001, Toulouse, France (2001) 38–44
8. Bigert, J., Ericson, L., Solis, A.: Missplel and AutoEval: Two generic tools for automatic evaluation. In: Proceedings of Nodalida 2003, Reykjavik, Iceland (2003)

9. Knutsson, O., Bigert, J., Kann, V.: A robust shallow parser for Swedish. In: Proceedings of Nodalida 2003, Reykjavik, Iceland (2003)
10. Voutilainen, A.: Parsing Swedish. In: Proceedings of Nodalida 2001, Uppsala, Sweden (2001)
11. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of IWPT 2003, Nancy, France (2003) 149–160
12. Nivre, J., Hall, J., Nilsson, J.: Memory-based dependency parsing. In: Proceedings of CoNLL, Boston, MA (2004)
13. Megyesi, B.: Data-Driven Syntactic Analysis – Methods and Applications for Swedish. PhD thesis, KTH, Stockholm, Sweden (2002)
14. Ejerhed, E., Källgren, G., Wennstedt, O., Åström, M.: The Linguistic Annotation System of the Stockholm-Umeå Project. Department of Linguistics, University of Umeå, Sweden (1992)
15. Brants, T.: TnT – a statistical part-of-speech tagger. In: Proceedings of ANLP 2000, Seattle, USA (2000)