



WHY YOUR WORD PROCESSING SOFTWARE DOESN'T FIND SPELLING AND GRAMMAR ERRORS VERY WELL...AND NEVER WILL!

By Anna Rumshisky

Ms. Rumshisky studied Applied and Theoretical Linguistics at Moscow State University in Russia and Cognitive and Computational Linguistics at Brandeis University. She is currently working toward a Ph.D. in Computer Science at Northeastern University College of Computer Science. She has developed software for automated text processing, including modules such as a part-of-speech tagger, a lemmatizer, parser, an entity-extractor, an anaphoric relations identifier, etc. She has also worked with and adapted for the purposes of statistical NLP research a number of different text processing tools, including rule-based and statistical parsers, machine-readable dictionaries and lexical databases.

For further information, please refer to the ProofreadNOW.com Business Plan, available from Phil Jamieson at PJamieson@ProofreadNOW.com.

1.0 Existing Grammar Checkers: A Sample Survey of Performance

During proofreading, a trained native speaker of English performs a number of different correctness checks. In addition to misspellings and typographical errors in punctuation, a proofreader routinely looks out for unfortunate wording, checks the semantic cohesiveness of each sentence, as well as text as a whole, assesses consistency of style, and so on. Can automated grammar checkers give a comparable performance? Grammar checkers are currently being provided with a variety of commercial software products, both with text-editing tools and specialized language applications, such as OCR software. The word processors specifically are often claimed to be capable of checking both grammar and style of writing.

There are currently two main types of automated grammar and style checkers. The first uses a grammar-based approach that attempts to produce a parse for each sentence. The second relies instead on pattern matching, sidestepping grammatical analysis. Grammar-based grammar checkers are included, for example, with Corel WordPerfect (based on Grammatik grammar checker by Reference Software), with Microsoft Word (based on Houghton Mifflin's CorrecText grammar correction system), etc. Some popular standalone programs, such as WinProof (a version of PC Proof by Intellect Systems), also use the grammar-based approach. Other grammar checkers, such as StyleWriter by Editor Software, use the strategy based on pattern matching. It has been argued, however, that the future of automated grammar checking belongs to the modular grammar-based strategy (see, for example, Cole et al., 1996; Oliva, 1995; Sagvall Hein, 1998).

As is well known, all currently available grammar checkers suffer from the so-called false flagging problem, that is, they mistakenly identify grammatically correct sentences as erroneous (cf. EAGLES, 1996; Wampler, 1995). At the same time, incorrect constructions are not detected. To illustrate, let us give a few sample tests of the grammar checker provided with Microsoft Word 98. Since all higher-level processing (i.e. stylistic checks, punctuation corrections, etc.) is based on grammatical analysis, we will test it on some straightforward parsing tasks.

1.1 Testing Grammatical Analysis: Examples

Let us first test how it fares with simple agreement. In the examples below, the ungrammatical constructions are marked (by this human writer!) with asterisks. The constructions indicated as ungrammatical by Microsoft Word 98 are underlined.

- The girls love this boy.
- * The girls loves this boy.
- * The girls dearly loves the boy.

However, just how sophisticated is the back-end processing that allows these sentences to be recognized as ungrammatical? Let us check what happens when a full parse is required to detect ungrammaticality.

- The boy the girls love is here.
- * The boy the girls loves is here.
- * The boy the girls love are here.
- * The boy the girl love is here.
- * The boy the girls loves are here.

It is easy to see that this grammar checker does reasonably well only with examples where ungrammaticality is localized. Thus, in the third sentence, it clearly doesn't recognize the word "love" as a verb, and indicates that there is lack of grammatical agreement between the words "love" and "are." For the same reason, the ungrammaticality is not detected at all in the fourth and fifth sentences.

Now, let us test this grammar checker on a couple of very common errors that typists make, namely, using "it's" instead of "its", or "to" instead of "too."

- It was too easy.
- * It was to easy.
- * They returned to quickly.
- * They returned too quickly finish the job.

- The ship has completed its voyages.
- * The ship has completed it's voyages.
- * The ship has completed all it's voyages.

It seems to do fairly well with cases that require only the knowledge of part-of-speech category of the neighboring word, but when that information is insufficient, it fails to perform the processing necessary to resolve the ambiguity.

2.0 Current State of Automated Language Processing and Perspectives

The obvious question here is, first, why do the existing grammar checkers perform so poorly, and second, is it likely that their performance will dramatically improve in the near future? To answer that, we have to consider the type of processing required to approach the human level of text comprehension.

Before a human reader can understand a given sentence or a sequence of sentences and then assess its correctness, the input text goes through several layers of processing. With an inborn ease, the human reader manages to identify which words are grouped together and form phrases, identify phrases and attributes that refer to the same entity, assess the meaning of each composite phrase, and finally, compute the meaning of the sentence. As native speakers, we also easily keep track of all inter-sentential linkages, that is, we identify references to the same entity throughout the whole text. This includes keeping track of pronouns, named entities, and phrases with definitive articles, etc.

2.1 Part-of-Speech Tagging and Parsing

In trying to mimic this level of sophistication, the current state of the art in automated language processing normally includes the following steps. The text is first *tokenized*, or split into words and punctuation. It is then run through a part-of-speech tagger that identifies each token as a particular part of speech. The output of the tagger allows a parser to use rules or probabilistic weights to guess at which of the tokens fall together to form phrases. Thus, if the text consists of a phrase, "flies like a flower", the tagger might, depending on the context, tag the first word as a noun in plural, and the second as a verb in present tense. Alternatively, it might tag the first word as a verb in third person singular form, and the second as an adverb.

The taggers currently considered most efficient are the probabilistic taggers, such as stochastic Markov model taggers, or transformation-based Brill taggers. The accuracy of such taggers is reasonably high, getting up to 96-97% of words tagged correctly (Charniak et al., 1993; Jurafsky & Martin, 1998). However, in order to perform best, these taggers need to be trained to a particular type of text. For example, in order to tag financial news articles, the probabilities they use when assigning tags must be obtained on a corpus of similar texts. As one switches from newspaper columns to technical reports to journal articles, the precision drops considerably, unless the taggers are retrained.

The output of the tagger is run through a parser. Automated parsers do not perform nearly as well as parts-of-speech taggers. An inaccurate parse means that words that do not in fact form a phrase are grouped together. In the recent 5 or 6 years, as statistical parsers were introduced in place of rule-based ones, there has been a perceptible jump in the accuracy. However, even top of the line statistical parsers still produce rather poor figures. The current ballpark of average accuracy is 80-85% (Collins, 1997; Charniak, 1999). In practical terms, it means that even for the texts of the same type, on which a statistical parser has been trained, one out of every 5-6 identified phrases will contain words that do not belong together. That is, every sentence with a non-trivial syntax will have one or more misparses.

The automated parsers are notoriously known to stumble on such common problems as prepositional phrase attachment, identifying clauses correctly within a sentence, identifying conjoined phrases, correctly segmenting noun compounds, recognizing a phrase whose integrity is indicated by agreement, rather than proximity within the sentence. None of these presents problems for a reasonable human reader. In case of conjunction, for example, imagine a trivially

ungrammatical sentence, "Your blue dress and ribbons doesn't go together." An automated text processor will most likely simply fail to group together the words "your blue dress and ribbons." Consequently, it will not recognize the phrase as the subject of the sentence requiring a verb in plural form.

Now we can see what actually happens when an error such as using "it's" instead of "its", or "to" instead of "too" is encountered in the text. The spell-checker does not catch it, of course, since both versions are in the dictionary. So at the next level of processing, a part-of-speech tagger will mistag the word in question. Additionally, stochastic taggers depend on the probability distribution of part-of-speech categories for adjacent words. So the tagger will most likely tag not only the word in question incorrectly, but also the surrounding words. At this point, checking the sentence for correctness of simple grammar, without even considering semantic well-formedness, becomes all but impossible.

2.2 Knowledge Base Survey

The current consensus in the field of automated text processing seems to be that the precision obtained with statistical parsers is not likely to improve, unless statistical processing techniques are augmented with lexical information (see, for example, Resnik, 1998; Manning & Shütze, 1999; Lauer, 1995; Mahesh & Nirenburg, 1996).

Lexical information, i.e. the information about the semantics of the words, is what allows a native speaker to resolve many of the attachment ambiguities in parsing and identify the correct meanings of phrase components, allows to compute the meanings of individual sentences, and to keep track of the entities referred to by different phrases throughout the whole text. While there is some disagreement about whether it is generic 'world knowledge,' or the language-specific information about word meanings that allows us to do that, it is not a matter of debate that no automated language processing system can succeed fully without a comprehensive knowledge base.

Several attempts have been made to create knowledge ontology on a large scale (Fellbaum, 1998; Alshawi, 1992; Lenat, 1995). However, all of them so far ended up being of limited use for practical purposes (cf. Mahesh & Nirenburg, 1996; Lenat et al., 1995). In order to be productively used in an automated text processing system, world knowledge and/or lexical information must be formally represented in a coherent all-encompassing manner. At the same time, this representation must be flexible enough to allow for such age-old conundrums as resolution of word ambiguity. In the current state of the field it appears all but impossible to satisfy both of these conditions on a large-scale segment of vocabulary.

3.0 Conclusions

Given this state of affairs in automated text processing, it becomes very apparent how inadequate the existing means in fact are. Automated grammar checkers can clearly help to eliminate human error by assisting with easily detectable errors. However, even with the current advances in statistical methods, it is highly improbable that the level of sophistication necessary to substitute a human proofreader will be achieved in the foreseeable future, if, indeed, at all. Much more likely, automated text processing will be developed further for highly specialized domains, those for which the task of formalizing what is called 'world knowledge' might be at least partially resolved.

REFERENCES

- Alshawi, H., ed.. (1992). The Core Language Engine. Cambridge, MA: MIT Press.
- Charniak, E., Hendrickson, C., Jacobson, N. & Perkwitz, M. (1993). Equations for Part-of-Speech Tagging. In Eleventh National Conference on Artificial Intelligence, (pp. 784-789). Washington, DC: AAAI Press/The MIT Press.
- Charniak, E. (1999). A Maximum-Entropy-Inspired Parser. Technical Report CS99-12 Department of Computer Science, Brown University.
- Cole, R. A., Mariani, J., Uszkoriet, H., Zaenen, A., and Zue, V., eds. (1996). Survey of the State of the Art in Human Language Technology. Cambridge University Press, Stanford University, Stanford, CA.
- Collins, M. (1997). Three Generative, Lexicalised Models for Statistical Parsing. In Proceedings of the 35th Annual Meeting of the ACL.
- EAGLES (Expert Advisory Group on Language Engineering Standards). (1996). Evaluation of Natural Language Processing Systems, EAG-EWG-PR.2, ILC-CNR, Pisa
- Fellbaum, C., ed. (1998). WordNet: An Electronic Lexical Database. Cambridge, MA: The MIT Press.
- Jurafsky, D. & Martin, J. S. (1999). Speech and Language Processing: An Introduction to Speech Recognition, Computational Linguistics, and Natural Language Processing. Draft of January 1999.
- Lauer, M. (1995). Designing Statistical Language Learners: Experiments on Noun Compounds. (Ph.D. Thesis). Macquarie University, Australia.
- Lenat, D., Miller, G. & Yokoi, T. (1995). CYC, WordNet, and EDR: Critiques and Responses. Communications of the ACM, 38(11), 45-48.
- Lenat, D. (1995). CYC: A Large-Scale Investment in Knowledge Infrastructure. Communications of ACM, 38(11), 33-38.
- Mahesh, K. & Nirenburg, S. (1996). Knowledge-Based Systems for Natural Language Processing. NMSU CRL MCCA-96-296.
- Manning, C., Schuetze. H. (1999). Foundations of Statistical Natural Language Processing. Cambridge, Massachusetts: The MIT Press
- Oliva, K. (1995). Grammar-Based Grammar Checker. A feasible practical implementation of high-level language technologies. In Proceedings of the 5th German Student Conference on Computational Linguistics (invited contribution), Saarbr_cken
- Resnik, P. (1998). WordNet and class-based probabilities. In WordNet: An Electronic Lexical Database, edited by Chris Fellbaum. Cambridge, MA: The MIT Press.
- Sågvall Hein, A. (1998). A Chart-Based Framework for Grammar Checking. In Proceedings of The 11th Nordic Conference on Computational Linguistics. University of Copenhagen.
- Wampler, Bruce E. (1995). Risks of grammar checkers. In Forum on Risks to the Public in Computers and Related Systems ACM Committee on Computers and Public Policy, 17(54).