Chapter 19

Mining for parsing failures

Daniël de Kok
University of Groningen

Gertjan van Noord
University of Groningen

Error mining is a technique to support the engineering of natural language parsers, by analysing parser output for a large set of inputs. It produces a list of properties (such as words or word sequences) of inputs which systematically lead to parsing failure. These properties typically point at omissions and mistakes in the grammar or the lexicon. Error mining can be applied to improve general purpose parsers, but is particularly suited to adapt parsers for novel text genres and topic domains.

In this article, a new error mining method is described, generalizing and extending earlier proposals by van Noord (2004), Sagot & de la Clergerie (2006) and de Kok, Ma & van Noord (2009). The new method improves the extraction of longer word sequences as features for the miner in comparison with the method of van Noord (2004), integrating the computation of suspicion as proposed by Sagot & de la Clergerie (2006). The extension allows the possibility to mine with character sequences as opposed to word sequences.

The new error miner is evaluated both quantitatively and qualitatively, and is shown to perform better than its predecessors.

1 Introduction

An important aspect of the engineering of natural language processing applications involves quality control. It is important to know for which types of input the result of the parser is not reliable. If we adapt the parser, it is important to check that the parser does not make new mistakes. And if we want to apply the parser for a novel text genre or topic domain, it is important to see what problems the parser faces in new contexts.

The importance of quality control has been recognized early on, and one of the first initiatives to do this in a systematic way has been pioneered by John Nerbonne and his collegues (Nerbonne et al. 1993). They manually built a large catalogue of

example sentences displaying the relevant syntactic properties of German, with the explicit purpose of testing NLP systems. Treebanks constitute obvious resources for quality control but likewise involve manual labour.

A technique which does not rely on manual construction of example sentences or manual annotation of corpus sentences is *error mining*. The goal of error mining is to discover properties of input sentences which systematically cause parsing failure. The parser is applied to a large amount of sentences. For each sentence, we record (or estimate) whether the parse was successful. The error miner then lists properties which occur frequently in sentences that cannot be parsed, and which hardly if ever occur in sentences that were unproblematic.

Error mining thus requires that we know whether a parse for a given input is successful. Obviously, for previously unseen sentences we do not know the quality of the output of the parser. Instead, therefore, for hand written grammars and lexicons, we use a weaker notion of success. A parse is deemed successful if the parser is able to construct a parse tree dominating all words of the sentence. In other situations, there may be alternative possibilities to estimate parse success. If the parser produces a confidence score as part of the output, we may want to use that score as an indication of parse success. In the formalization below, we assume that parse success is not necessarily a binary distinction, but a score between 0 and 1.

The result of error mining consists of superficial properties of input sentences which 'cause' parsing failure. For instance, one such property could be the bigram "`why .`". This indicates that sentences in which the word `why` is immediately followed by a full stop are typically not parsed correctly. The parser engineer will immediately realize (perhaps after further inspection of all sentences which contain this bigram) that verbs which introduce indirect questions should also be able to combine with bare WH-phrases, as in *They don't know why.*

In this paper we review the earlier approaches to error mining by van Noord (2004) and Sagot & de la Clergerie (2006) (section 2). Extending work we described earlier (de Kok, Ma & van Noord 2009), we present a generalized error miner which combines the strength of these two former proposals (section 3).

In section 4 we discuss a simple evaluation method based on precision and recall. We apply this method to the error miners. The comparison shows that the generalized error miner out-performs the former proposals.

Error mining has been used for several parsing systems and several languages, including English, Dutch, French, German and Spanish Error mining has also been applied to improve natural language generation (Gardent & Narayan 2012; Narayan & Gardent 2012).

2 Previous work

Two promising error mining techniques have been proposed. van Noord (2004) uses word n-grams of arbitrary length as its features and implements a simple, frequency-based suspicion scoring method. Sagot & de la Clergerie (2006) proposes a more sophisticated iterative suspicion scoring method. However, that method provides

only rudimentary feature extraction, since it only considers words and word pairs.

Our contribution in this paper is to combine an improved version of the more general feature extraction method of van Noord (2004) with the successful suspicion computation of Sagot & de la Clergerie (2006).

In the following description we use an error function error(s), which is zero if the sentence s was parsable or one if it was not parsable. The suspicion of the feature f_i, susp(f_i), is the mean error score of the sentences in which the feature f_i occurs. Here, S is the bag of sentences, and $f_i(s)$ is the number of occurrences of feature f_i in sentence s.

$$\text{susp}(f_i) = \frac{\sum_{s \in S} f_i(s) \cdot \text{error}(s)}{\sum_{s \in S} f_i(s)} \tag{1}$$

If word n-grams are used as features, a potential problem arises. If a particular word sequence $w_i \dots w_j$ has a high suspicion, then all longer word sequences which contain $w_i \dots w_j$ will necessarily have a high suspicion too. This is undesirable. Therefore, if n-grams are included as features in the error miner, a selection criterion is required. van Noord (2004) proposes to add a longer n-gram only if its suspicion is higher than all of the n-grams it contains:

$$\text{susp}(w_h \dots w_i \dots w_j \dots w_k) > \text{susp}(w_i \dots w_j) \tag{2}$$

As a result, there usually is only a small number of long n-grams which the error miner needs to take into account. This also implies that there is no need to set a value for n a priori; instead, the data determines which longer n-grams are required. As a further heuristic, if a longer n-gram satisfies the selection criterion, then the corresponding shorter n-grams are no longer used as features for the error miner.

The error mining method described by Sagot & de la Clergerie (2006) addresses an issue in the miner of van Noord (2004) where features that co-occur frequently with problematic features also get a high suspicion. It does so by taking the following characteristics of suspicious features into account during feature selection: if a feature occurs in parsable sentences, it becomes less likely that it is the cause of a parsing error; the suspicion of a feature depends on the suspicions of other features in the sentences where it occurs; a feature observed in a shorter sentence is initially more suspicious than a feature observed in a longer sentence.

The method introduces the notion of *observation suspicion*, susp($f_i(s)$) which is the suspicion of feature f_i in sentence s. The (global) suspicion of a feature is the average of all observation suspicions,

The observation suspicions are dependent on the feature suspicions, making the method iterative. The observation suspicion is defined as the feature suspicion, normalized by suspicions of the other features that are active within the same sentence:

$$\text{susp}^{n+1}(f_i(s)) = \frac{\text{error}(s) \cdot \text{susp}^{n+1}(f_i)}{\sum_{f_j \in F(s)} \text{susp}^{n+1}(f_j)} \tag{3}$$

Here, $F(s)$ is the set of features that are active (have a non-zero frequency) in sentence s. Since the mining procedure is iterative, the suspicion of a feature is redefined to depend on the observation suspicions of the previous iteration:

$$\text{susp}^{n+1}(f_i) = \frac{\sum_{s \in S} \text{susp}^n(f_i(s))}{\sum_{s \in S} f_i(s)} \tag{4}$$

Given the recursive dependence between feature suspicions and observation suspicions, starting and stopping conditions are defined for iterative mining. The observation suspicions are initialized by uniformly distributing suspicion over the features that are observed in a sentence:

$$\text{susp}^0(f_i(s)) = \frac{\text{error}(s) \cdot f_i(s)}{\sum_{f_j \in F(s)} f_j(s)} \tag{5}$$

Mining is stopped when the process reaches a fixed point where the feature suspicions have stabilized.

3 n-gram expansion

While the iterative miner described by Sagot & de la Clergerie (2006) only mines on features that are word unigrams and bigrams, our experience with the miner described by van Noord (2004) has shown that including n-grams that are longer than bigrams as features during mining can capture many additional phenomena.

We propose a feature extraction method that adds and expands n-grams when it is deemed useful. This method iterates through a sentence unigram by unigram and expands unigrams to longer n-grams when there is sufficient evidence that the expansion will be useful. We then combine this feature extractor with the selection method of Sagot & de la Clergerie (2006). Within this extractor, we use the definition of suspicion given as Equation 1, except that we do not use the frequency $f_i(s)$ directly, but use a binary value which indicates if the form f_i occurs in the sentence s.

The expansion method is based on the following observation: if we consider the word bigram w_1, w_2, either one of the unigrams or the bigram can be problematic. If one of the unigrams is problematic, the bigram will inherit suspicion of the problematic unigram. If the bigram is problematic, the bigram will have a higher suspicion than both of its unigrams. Consequently, we will want to expand the unigram w_1 to the bigram w_1, w_2 if the bigram is more suspicious than both of its unigrams. If the bigram is just as suspicious as one of its unigrams, such an expansion is not necessary, since we want to point to the cause of the parsing error as exactly as possible.

The same procedure is applied to longer n-grams. For instance, the expansion of the bigram w_1, w_2 to the trigram w_1, w_2, w_3 is only permitted if w_1, w_2, w_3 is more suspicious than its bigrams. Given that the suspicion of w_3 aggregates to w_2, w_3, we account for w_3 and w_2, w_3 simultaneously in this comparison.

The general criterion is that the expansion to an n-gram $i \ldots j$ is permitted when susp$(i \ldots j) >$ susp$(i \ldots j-1)$ and susp$(i \ldots j) >$ susp$(i+1 \ldots j)$.

This method differs from that of van Noord (2004) in that the method of van Noord (2004) considers all n-grams in a sentence, while our method does not consider $w_i \ldots w_j \ldots w_k$ if the expansion to $w_i \ldots w_j$ failed.

In Table 1, we illustrate our method to expand the n-gram feature *voor* to *voor uur van de* in the following sentence:

(6) De Disney-topman staat voor uur van de waarheid.
the Disney top executive stands before hour of the truth.
'The moment of truth has come for the Disney top executive.'

The counts in this example are based on real data.

Table 1: Expansion of the feature *voor* to *voor uur van*.

Expansion	susp$(i \ldots j)$	susp$(i \ldots j-1)$	susp$(i+1 \ldots j)$	**Expand**
voor $\rightarrow$ *voor uur*	$\frac{48}{50}$	$\frac{778949}{9590152}$	$\frac{116975}{1563498}$	yes
$\rightarrow$ *voor uur van*	$\frac{40}{40}$	$\frac{48}{50}$	$\frac{856}{9779}$	yes
$\rightarrow$ *voor uur van de*	$\frac{30}{30}$	$\frac{40}{40}$	$\frac{297}{3748}$	no

While this expansion method looked promising in our initial experiments, we found it to be too eager. This eagerness is caused by sparsity in the data. Since longer n-grams are less frequent, they also tend to be more suspicious if they occur in unparsable sentences. The expansion criterion does not take data sparseness into account.

We introduce an expansion factor to handle sparseness. This factor depends on the frequency of an n-gram in the set of unparsable sentences, and asymptotically approaches one for higher frequencies. As a result the burden of proof is inflicted on the expansion: the longer n-gram needs to be relatively frequent or much more suspicious than its (n-1)-grams. The expansion criteria are changed to susp$(i \ldots j) >$ susp$(i \ldots j-1) \cdot extFactor(i \ldots j)$ and susp$(i \ldots j) >$ susp$(i+1 \ldots j) \cdot extFactor(i \ldots j)$, where

$$extFactor(i \ldots j) = 1 + exp(-\alpha \sum_{s \in S} \text{error}(s) \cdot f_{i \ldots j}) \tag{7}$$

As we show in section 4.5, $\alpha = 0.01$ proved to be a good setting.

After error mining, we can extract a list of forms, ordered by suspicion. However, normally we are interested in forms that are both suspicious and frequent.

4 Evaluation

4.1 Methodology

In the early papers on error mining, error mining methods have been evaluated primarily manually. Both van Noord (2004) and Sagot & de la Clergerie (2006) conduct a qualitative analysis of highly suspicious features. But once one starts experimenting with various extensions, such as n-gram expansion and expansion factor functions, it is difficult to gauge the contribution of modifications through a small-scale qualitative analysis.

To be able to evaluate changes to the error miner, we have supplemented qualitative analysis with a automatic quantitative evaluation method. Such a method should judge an error miner in line with the interests of a grammar engineer:

- an error miner should return features that point to problems that occur in a large number of sentences;
- the features that are returned by the error miner should be as exact as possible in pointing to the problem.

The first requirement is relatively easy to test — the error miner should return features that occur in a relatively large number of unparsable sentences. It is less clear how the second requirement should be tested, since it requires that a human checks the features to be relevant. However, if we assume that the quality of features correlates strongly to their discriminative power, then we would expect a miner to return features that only occur in unparsable sentences. These characteristics can be measured using the recall and precision metrics from information retrieval:

$$R = \frac{|\{S_{unparsable}\} \cap \{S_{retrieved}\}|}{|\{S_{unparsable}\}|} \tag{8}$$

$$P = \frac{|\{S_{unparsable}\} \cap \{S_{retrieved}\}|}{|\{S_{retrieved}\}|} \tag{9}$$

Consequently, we can also calculate the f-score (we use $\beta = 0.5$).

4.2 Material

In our experiments, we use the Flemish Mediargus newspaper corpus. This corpus consists of 67 million sentences (1.1 billion words). For 9.2% of the sentences no full analysis could be found. Flemish is a variation of Dutch written and spoken in Belgium, with a grammar and lexicon that deviates slightly from standard Dutch. Previously, the Alpino grammar and lexicon were never systematically modified for parsing Flemish.

We now proceed to discuss the results of the quantitative and qualitative evaluation. We will first compare the miners described in van Noord (2004) and Sagot & de la Clergerie (2006). Then, we examine the performance of the expansion method

that we proposed and compare it to the competition. Finally, we will conclude this section with an qualitative evaluation of iterative error mining with our expansion method.

4.3 Scoring function

After error mining, we can extract a list of forms, ordered by suspicion. However, normally we are interested in forms that are both suspicious and frequent.

$$delta(f_i) = \sum_{s \in S, \text{error}(s) > 0} f_i(s) \; - \sum_{s \in S, error = 0} f_i(s) \tag{10}$$

$$\text{score}(f_i) = \; \text{susp}(f_i) \cdot delta(f_i) \tag{11}$$

$$\text{score}(f_i) = \begin{cases} 0 & \text{if } delta(f_i) \leq 0 \\ \text{susp}(f_i) \cdot (1 + ln(delta(f_i))) & \text{if } delta(f_i) > 0 \end{cases} \tag{12}$$

We use the scoring function that performed best for a specific error miner. In the case of the iterative miner of Sagot & de la Clergerie (2006) and the miner proposed in this work, the scoring function in equation 11 is used in the experiments below. For the miner of van Noord (2006), the scoring function in equation 12 is used below.

4.4 Iterative error mining

Our first interest is if, and how much iterative error mining outperforms error mining with suspicion as a ratio. To test this, we compared the method described by van Noord (2004) and the iterative error miner of Sagot & de la Clergerie (2006). For the iterative error miner we included all bigrams and unigrams without further selection. The left graph in Figure 1 shows the F-scores for these miners after N retrieved features.

The iterative miner of Sagot & de la Clergerie (2006) clearly outperforms the miner of van Noord (2004), despite the fact that the latter has a more sophisticated feature extraction method. That this happens is understandable — suppose that 60% of the occurrences of a frequent feature is in unparsable sentences. In such a case, the ratio-based miner would assign a suspicion of 0.6. But, since the feature is relatively frequent, it would still be ranked very high, even though there is plenty of evidence that it is not responsible for parsing errors. This also manifests itself in the performance of scoring functions — the ratio-based miner was the only miner to perform best with the scoring function in equation 12. This indicates that relying too much on frequencies is dangerous in ratio-based mining. However, relying purely on suspicion would return many forms with a low frequency.

Another interesting characteristic of these results is that the performance of the error miners seems to fit a logarithmic function. This is not surprising, since it shows that there are some very frequent patterns indicating errors and a long tail of less frequent patterns indicating errors. The fact that there is a long tail of infrequent patterns does not make the task of the parser engineer hopeless. In fact, a single error in the parser will often surface in multiple patterns. As an example, consider the Dutch determiner `zo'n` ('such'). In standard Dutch, this determiner combines with a singular noun. In Flemish, the determiner can combine with plural nouns as well. That usage of the determiner `zo'n` gave rise to parsing errors. This particular error shows up in many patterns which occur high in the list of relevant features: `zo'n momenten; zo'n mensen; Op zo'n momenten; zo'n omstandigheden; zo'n zaken`... Generalizing patterns to include part-of-speech tags would be useful here.

4.5 n-gram expansion

In our second experiment, we compare the performance of iterative mining on uni- and bigrams with an iterative miner that uses the n-gram expansion algorithm that was described in section 3. We use $\alpha = 0.01$, which provides good performance across the board. In the second graph of Figure 1, we compare our miner that uses word n-gram expansion with the miner of Sagot & de la Clergerie (2006). We observe that our method for the inclusion of longer n-grams is beneficial to error mining.

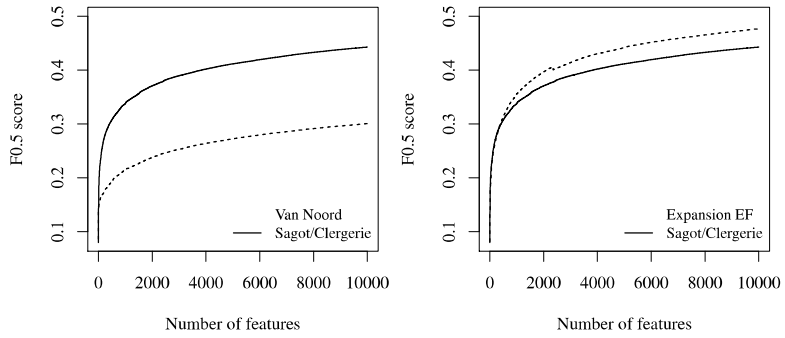

Figure 1: Left: $F_{0.5}$-scores after retrieving N features for ratio-based mining versus iterative mining with unigrams and bigrams. Right: $F_{0.5}$-scores after retrieving N features for the miner of Sagot and de la Clergerie (2006) and the miner proposed here

4.6 Further analysis

We found many interesting longer n-grams in the results of the miner proposed in this article that could not have been captured by the miner of Sagot & de la Clergerie

(2006). If we inspect the most relevant 50 items, using the relevance score function given in equation 11, we do not find any unigrams, 16 bigrams, and 34 longer N-grams.

One very instructive example from this list of 50 most relevant items is the trigram `Het zijn zij` ('It are they'). This example illustrates a convincing case were a bigram would not suffice. Each of the three words of the pattern, `het`, `zijn` and `zij`, are amongst the most frequent words in Dutch. Also, the bigrams `Het zijn` and `zijn zij` are very frequent, and not suspicious at all: the first bigram occurs 56947 times, including 6937 parsing failures; the second bigram occurs 5588 times (696 parsing failures). However, the trigram *is* very suspicious: it occurs 220 times, leading to parsing failure in 212 cases. The trigram is found in Flemish cleft sentences such as *Het zijn zij die de fouten maken* ('It is them who make the mistakes').

The pattern surfaces because in standard Dutch, in contrast to Flemish, the sentence would be phrased as *Zij zijn het die de fouten maken.*

5 Conclusions

We combined iterative error mining with a new feature extractor that includes n-grams of an arbitrary length, taking care that n-grams are long enough to capture interesting phenomena, but not longer. We dealt with the problem of data sparseness by introducing an expansion factor that softens when the expanded feature is very frequent.

In addition to the generalization of iterative error mining, we have introduced a method for automatic evaluation that is based on the precision and recall scores commonly used in information retrieval. This allows us to test modifications to error minings quickly without going through the tedious task of ranking and judging the results manually.

Using this automatic evaluation method, we have shown that iterative error mining improves upon ratio-based error mining. We have also shown that the use of a smart feature extraction method improves error miners substantially. The inclusion of longer n-grams captures many interesting problems that could not be found if a miner restricted itself to words and word bigrams.

References

de Kok, Daniël, Jianqiang Ma & Gertjan van Noord. 2009. A generalized method for iterative error mining in parsing results. In *GEAF*.

Gardent, Claire & Shashi Narayan. 2012. Error mining on dependency trees. In *ACL 2012*. http://dl.acm.org/citation.cfm?id=2390524.2390607.

Narayan, Shashi & Claire Gardent. 2012. Error mining with suspicion trees: seeing the forest for the trees. In *COLING 2012*.

Nerbonne, John, Klaus Netter, Abdel Kader Diagne, Judith Klein & Ludwig Dickmann. 1993. A diagnostic tool for German syntax. *Machine Translation* 8(1). 85–107.

Sagot, Benoît & Éric de la Clergerie. 2006. Error mining in parsing results. In *ACL-44: proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 329–336. Sydney, Australia: Association for Computational Linguistics.

van Noord, Gertjan. 2004. Error mining for wide-coverage grammar engineering. In *ACL*, 446. Barcelona, Spain: Association for Computational Linguistics.

van Noord, Gertjan. 2006. **A**t **L**ast **P**arsing **I**s **N**ow **O**perational. In *TALN 2006 Verbum Ex Machina, actes de la 13e Conference sur le Traitement Automatique des Langues naturelles*, 20–42. Leuven.