Learning and Application of Differential Grammars

David M. W. Powers
Artificial Intelligence Laboratory
Department of Computer Science
The Flinders University of South Australia
powers@acm.org

Abstract

We examine the Differential Grammar, a representation designed to discriminate which of a set of confusable alternatives is most likely in the context it occurs in. This approach is useful whereever uncertainty may exist about the identity of a token or sequence of tokens, including in speech recognition, optical character recognition and machine translation. In this paper our application is word processing: we discuss multiple models of confusion which may be used in the identification of confused words, we show how significant contexts may be identified and condensed into Differential Grammars, and we contrast the performance of our implementation with that of two commercial grammar checkers which purport to handle the confused word problem.

1 Introduction

In this paper, we explore the concept of a Differential Grammar and apply it to the problem of grammar checking – in the sense that it is used on the box your WYSIWYG word processor came in! A Differential Grammar is not a grammar in the traditional rule-oriented sense, and although it is lexically-focussed it doesn't really have a concept of a rule at all, but is somewhat more specialized, and extremely simple. We introduce a simple statistical preclassification, which we use to define a modified Ngram environment for each member of a confusion class, and then adjust the size of our environment until a predetermined significance level or diameter is reached. We also demonstrate several approaches to the automated generation of the confusion classes.

The structure of the paper is as follows: we introduce Differential Grammars and motivate them in terms of grammar checking, we discuss the acquisition, efficiency and significance issues, we present an improved user interface for grammar checkers, and we demonstrate the improvement achieved by Differential Grammar based checkers compared with commercial products. Finally, we discuss other experiments relating to automated acquisition of differential grammars.

2 Commonly Confused Words

A word processor's grammar checker typically checks grammar in two senses, a pre-/proscriptive sense which should be characterized by what are more properly identified as style rules, and a spell-checking like sense which is characterized as the use of grammar rules to identify typos, substitutions, omissions, duplications which occur at the word or letter level.

The errors we focus on are those 'typos' which result in one word being substituted by another word – both of which are words which would be accepted by the spell-checker. This is closely related to the problems of commonly confused words, homophones and near homophones, but the most important difference between the different types of substitution is in fact whether readers, proofreading their own work, or looking at a detected error, will recognize that it is an error or not.

This is not a new problem, either from the point of view of commercial word processors (Johnson, 1992) or Computational Linguistics (Ill, 1983), and two aspects to the problem must be differentiated: we want to find the typos, but we don't want to be overwhelmed with false errors. It is this latter concern which leads to people deciding not to use the available grammar checkers – and those who do tend to turn off the style rules, which they regard as pure noise, making the whole question of the value of grammar checkers rather controversial (Wampler, 1995).

David M. W. Powers (1997) **Learning and Application of Differential Grammars.** In T.M. Ellison (ed.) *CoNLL97: Computational Natural Language Learning,* ACL pp 88-96.

By targeting commonly confused words, rather the general problem, we have simply to search for contexts which differentiate the words well. Furthermore, we only wish to distinguish words on grammatical grounds – probabilistic methods which rely on semantic associations are simply too sensitive to changes in genre or topic to use as the primary discriminant.

3 Differential Grammars

A differential grammar is basically a small set of environments that allows us to differentiates between a pair of confused words in all contexts(Kernick, 1996). However, we want to emphasize that the approach need not be limited to word confusion or grammar checking, and that there is no need to limit it to pairs of targets. Conversely, we also want to strengthen the definition slightly, as we want to syntactic errors. We therefore define a Differential Grammar as:

Definition: Differential Grammar

> A minimal set of syntactically significant environments that differentiate amongst a set of possible targets.

However, we do not wish to have to specify the differential grammars or the syntactic environments, but rather wish to learn them. For the commonly confused word problem (in the more general and inclusive sense that encompasses everything from common typos to near homophones), we thus have three aspects to the problem where we wish to do some kind of learning:

1. identifying pairs of commonly confused words;
2. selecting appropriate syntactically significant environments;
3. deciding when an error has occurred.

This involves learning in three different senses, and the programs we present here have performed each kind of learning to varying degrees. First, we want to learn to select appropriate data for applying our grammar building methodology to – we want to avoid having to provide positive and negative examples. Second, we want to learn what is syntactically appropriate – we want to avoid having to provide tags, bracketing or parses. Third, we want to learn and dynamically adjust to what the user wants and the target text requires – we want to avoid users having to set parameters.

In addition we have some further goals relating to optimization:

4. minimizing the size of the differential grammar;
5. ensuring the significance of the contexts stored;
6. facilitating the users' interactions with the system.

4 Discovery of Confused Word pairs

We consider here the first of our six goals. We assume that certain substitutions are more likely than others, and we do not aim to deal with the general case which includes arbitrary unlikely substitutions. We distinguish six different types of reasons for substituted word errors:

a. typos: keyboard proximity (knowledge of keyboard used);
b. phonos: phonological proximity (phonological features used);
c. grammos: grammatical proximity (grammatical features used);
d. frequens: frequency disparity (frequency information used);
e. foreignish: interlinguistic disparity (not targeted at present);
f. idiosyncratic: unknown reason (some system or user confuses the pair).

Note that we do not include semantic or style errors - the latter tend to be a result of prescriptive linguists proscribing certain constructs, or maintaining traditional distinctions which are falling out of common use: e.g. the distinction between 'due to' (as meaning 'caused by' but not 'because of') and 'owing to' (as meaning 'because of' but not 'caused by'); rules about prepositions at the end of sentences; split infinitives; deprecated passives; etc.

In our first set of experiments (a) has been modeled by simple adjacency on the keyboard, testing in principle all pairs in decreasing order of frequency of the more frequent member. Errors are modeled primarily as systematic displacements of the hand on the keyboard, or substitutions of adjacent characters in the order 1 case. Deletions are handled by treating the empty string '' as being adjacent to all characters, and insertions are handled inversely as deletions. Transpositions and interspersions can be ranked on the number of moved characters, and displacements by the number of substituted characters, but in our experiment we limited both to one. The grammatical errors (c) are somewhat trickier to characterize, but a brute-force first approximation would simply list all morphological derivations from each root, ideally working at the morphological level.

In relation to (f) we note that for any confusion pair identified by the commercial grammar checkers on our test texts (true error or false error), we have forced generation of Differential Grammars. We are trying to target the exact same class of substitution errors. This has two effects: it increases the possibility of false errors and decreases the possibility of missed errors.

The frequens (d) are an interesting class (Kernick, 1996) in that we tend to make disproportionately more errors in which one word of the confused pair is very frequent and the other less so. The very frequent words seem to be more easily activated than their near homonyms, and we have tendency to type the frequens automatically even when it is the less frequent partner that was intended. This is handled in our experiments by our use of the higher of the two frequencies ranking for grammar generation and evaluation. In addition, we could (but don't) allow more latitude in the search for pairings of frequent words in classes (a), (b) and (c), e.g. by increasing the number of characters or features that might be out of place, substituted or inserted. Instead, we have manually included pairs involving some of the most common words.

Two common errors (which I made in typing the last paragraph) are 'out' → 'our' (a) and 'our' → 'are' (b/d), and the single most common error is 'its/it's' – but often the confusion classes are not simply pairs of words, e.g: 'yaw/your/yore/you're' and 'there/their/they're'.

We tested around 100 pairs of words generated automatically on the basis of keyboard proximity (a), as well as those proposed manually under (b) to (f). 76 were used in our system and 55 were rejected for lack of either significance or discrimination.

5 Building an efficient Differential Grammar

We now refine the concept of a differential grammar and present the specific form we employ. The first thing we need to do is to define what we mean by 'syntactically significant environments'. Basically we collect Ngram statistics for each of the target words, but we reduce the amount of statistics by a form of syntactic preclassification, then we start with a minimal diameter and progressively increase it until a desired degree of certainty and significance is reached.

We define an environment and its diameter straightforwardly (again generalizing (Kernick, 1996)):

Definition: Environment

A sequence of contiguous units which includes the target unit.

Definition: Diameter

The number of units other than the target which constitute the specified environment of the target unit.

We note that we have generalized the definition from the specific focus on words we have here: read 'word' for 'unit' throughout. Also we highlight the fact that there is not a unique environment for each target word, but rather that there are, in general, multiple environments for each possible diameter. Diameter 0 refers to the target word alone. It's frequency relative to the combined frequency of the other members of a confused word set provides a 0th order likelihood of the word being correct. However, because of the phenomenon of frequens, this is a highly unreliable estimate and we therefore eschew it no matter how great the order 0 likelihood of error may be. However, some commercial word-processors seem to use an order 0 model and flag all occurrences of certain words (Kernick, 1996).

We therefore look for significant environments of larger diameter and estimate the probability of each alternative in terms of their relative frequency in that environment. An environment of diameter N-1 corresponds to an N-gram and environment statistics are therefore derived from N-gram statistics. In practice we limit ourselves to near symmetric environments with the target word in the centre. This gives us a unique environment for each even diameter, but a pair of environments designated +N (more left context) and -N (more right context) for each odd diameter N.

Our algorithm stores statistics only for significant environments and increases the diameter progressively, up to a predefined maximum diameter or until a specified certainty threshold is reached. Note that we overload our term 'environment' to mean not only a specific environment of a target word in the current text, but the set of environments of the target word in the training corpus and having the specified diameter, as summarized in the collected statistics. Our usage will be clear from context, and will vary according to whether we are talking about testing/text or training/corpus (respectively).

This approach to the storing of Differential Grammars helps to keep the requirements for a given confused word set small, and thus contributes to our goal (4) of minimizing the size of the grammar, without significantly affecting reliability (precision and detection rate).

6 Significance & likehood estimates

We now discuss the different models of significance we have experimented with, and the issue of combining information from multiple environments. We will illustrate this with examples relating to the common substitution 'from' $\rightarrow$ 'form', and we will assume that a desired minimal level of significance s has been specified (0.95 in the examples). Note that models which are statistically significant are also likely to be skewed so as to provide good discrimination, but the reverse is not in general true. The more data we have, the more statistically significant a particular likelihood ratio is – so we also want to ensure 'significance' in a more application-oriented sense and only store information which is both significant and discriminative.

Since we are training on a large corpus, and allowing for potentially many confusion sets from a huge set of possibilities, we want to eliminate as quickly as possible those pairs which can't possibly be discriminated reliably. For this purpose we introduced a first order test based on the binary Laplacian estimator, and require that the number of instances of a specific environment N_d for the confusion set satisfies $N_d > 1/(1-s)-2$ (where d is its diameter). For our modest 95% significance level, only 18 examples are required. Note that, as previously discussed, we do not ever directly use the 0-diameter environment alone to determine likelihoods, but rather we insist that environments must be more significant than the 0-environment in terms of the Laplacian test.

Next we want to ensure that the each environment is not only significant in its own right but is significantly different from the next smaller environment. The example of 1105 instances of 'from' and 5 of 'form' being reduced to 47:1 (Kernick, 1996) is significant according to our Laplacian estimate, but the 1 could just have easily been a 0 or a 2 and thus doesn't improve on the smaller environment. The larger environment is not significantly different according to a likelihood derivative, which considers the rate of change of the ratio (Kernick, 1996), and the difference between the two environments is also not significant by Fisher's exact test (Winston, 1993) (or the closely related G^2 (Kilgarriff, 1996)) which assesses the probability that the distribution is due to chance.

Note that we make no attempt to correct or smooth data, since 0% and 100% likelihoods are not undesirable, and the 'corrections' are as likely to distort as to improve the data (Church and Gale, 1991), rather we discard data which does not reach significance. Our current model also discounts any cases handled by a larger environment (see section 9).

This leaves two further issues to discuss. The first is how we determine likelihoods. Normally, it is very simple, we take the biggest environment that matches and simply use the relative probability of the target word in the sampled environment. In the case of odd diameters, both a left and a right environment may exist and these may agree or disagree. Here we use the min operator to combine them – the minimum fits both the case where they tend to agree, and is conservative, or when they tend to differ, when if one side think it is wrong it is probably not correct. However, this is not always the case. An actual example from our experiment on Usenet text is: "I don't know where you're coming from on this" where the diameter one environments strongly suggest "coming from" and "form on" respectively. With a larger corpus a significant diameter two environment for 'from' would form to handle this idiom.

The second question is how to allow for the biasing effect of frequens, where we tend to make disproportionately more errors in the direction of the more frequent word, since this is also the direction our statistics are pointing. This effect however can have its impact reduced by taking larger environments and provides support for the intuitively and empirically determined threshold function of (Kernick, 1996), which converts the user supplied precision (ϕ) to a discrimination threshold value (θ) which increases as the diameter (d) of the environment reduces, thus giving more credence to the larger diameter environments: $\theta = \phi + (1-\phi)/2d$.

7 Focussing on syntactic contexts

Limiting our environments to a small diameter already biases toward correlations which express syntactic rather than semantic relationships (Finch, 1993), which very phenomenon is responsible for the success of Ngram models as an alternative to a grammar (Charniak, 1993). Semantic associations are normally found in larger segments in which the frequency of semantically primed words is higher than expected on the basis of relative frequency in the corpus or the language (this locally increased frequency has recently been dubbed 'clumping' (Church and Gale, 1995) and various measures have been proposed to compensate for it (Kilgarriff, 1996)).

Another statistical attribute which is associated with the syntax/semantics distinction is the raw frequency of a word. The most frequent words tend to be syntactic in nature, and will often function words or closed class words. The less frequent words tend to be more content words or open class words, like nouns and verbs (Kilgarriff, 1996). Due to the extreme skewing of the frequency distribution, varying inversely with rank according to Zipf's law, the first 150 words of a corpus with a lexicon of 250,000 words cover over half of the corpus (Kernick, 1996).

Collecting statistics based on these 150 'eigenwords', almost all of which are function words, gives us our a syntactic bias and we used the standard Unix list `/usr/lib/eign` in our initial experiments. Furthermore since the function words tend to act at fairly close quarters, these words are appropriate for smallish environments. However, we don't want our statistics to be restricted to environments composed solely of the 150 eigenwords, and we do not want to have to resort to just bigram statistics collected at various displacements (Finch, 1993). Indeed for function words, for anything but the smallest displacements, our experiments show that such bigram statistics quickly approach corpus/author norms (Kernick, 1996). The obvious step is to introduce an 'open class', denoted by 'O', as a placeholder for the rest of the lexicon. But we can to do better than this by finding other useful classes which are easy to discover using our collected statistics. We therefore move our search for syntactic cues to the morphological level. Again, rather than seeking to develop a formal morphology and associate grammatical information with the morphemes, we simply keep additional statistics for words classified by the most common affixes (we use 12 suffixes for English). Note that our residue class now represents the null morph, '0'.

After we include numbers and punctuation we end up with a nominal 172 'eigenunits', but irregular or problematic forms could usefully be added to reduce the noise in these blindly recognized classes, e.g, classes may have multiple syntactic functions ('-s' and the null morph '-0' can indicate a noun or a verb) and/or fortuitous mismatches ('-ed' and '-ing' will accept 'red' and 'ring').

Fortunately, such mismatches have a good chance of already being one of the 150 eigenwords (better than 50%) and a low probability of occurrence in any particular slot (a fraction of 1%). Those occurrences which are not systematic simply contribute to the overall noise in the method, whilst those which are systematic actually contribute to the success of the technique!

In addition, a broad definition of affix as a word-initial or -final sequence can give us affixes which may deviate from the morphological. In our first experiment we use only 12 hand-chosen suffixes, but in subsequent experiments we also split each of these classes according to whether they had a vowel or a consonant prefix, which permits us to ensure that we can deal with the 'a/an' distinction. Later we investigate how affixes may be discovered automatically. Note that (Entwisle and Groves, 1994) use essentially the same crude affix information to achieve complete parsing/validation of English sentences using a (computationally expensive) constraint parser.

This now allows us to complete the definition of our syntactic environment: The Ngram information is reduced to eigenunit environments by simply replacing each word other than the target by the first matching eigenunit (eigenwords are checked before affix classes, shortest first).

8 Interface

In addition to the learning of Differential Grammars using the pure syntactic methods defined above, and testing on 'known good' and 'expected bad' text, we have also paid some attention to the user interface. Two interfaces are available, an emacs interface – it works just like the spell checker – and a frame-based web-interface. We used the likelihoods to colour the words so that the words which are more likely to be wrong are highlighted more strongly. Also the environment used to make the decision is highlighted contrastingly. This is useful both for the user, and for the developer, in evaluating and enhancing the performance of the system.

We also allow the user to change the threshold at which notification of potential errors occurs. Normally this is set at a relative probability 0.75 for the target word relative to other members of the confusion class, a precision setting of 75% (our results are presented for this setting). If this threshold is exceeded the most likely replacement is automatically proposed.

9 Results

Table 1 presents results from an experiment using 100Meg of training text (TIPSTER, 1994) and three test texts of similar size but different character, in which Differential Grammars are trained and used to grammar check the test texts, which are also checked by two commercial systems. Our methodology is summarized generally in Fig. 1.

We trained Differential Grammars for 78 confusion pairs using 161 eigentokens and a 95% significance level and tested the grammar checker at the default 75% likelihood threshold. Performance was comparable with that of the two commercial systems, but all three systems showed individual coverage characteristics. The confusion pair 'its/it's' was responsible for our poorer performance on the newsgroup corpus (SFB), but we demonstrated that

GRAMMAR CHECKER	THC-f	THC-t	SFK-f	SFK-t	SFB-f	SFB-t	SFB-its
Microsoft Word	282	0	328	17	225	40	13
Word Perfect	75	0	116	21	77	49	9
Differential 75%	158	6	170	33	165	19	19

Table 1: *Initial results (Kernick, 1996) comparing three systems on three 12000 line texts of approximately 100000 words each. The -f columns represent* false *errors (lower if better) and the -t columns show* true *errors (higher is better). Our system could not resolve 'its/it's' which was the most common error in SFB so the final -its column shows the results with these errors discounted. The three corpora were chosen to be as similar as possible, including one published computer-related work (THC), science fiction genre text written by a member of our team (SFK), and text of the same genre taken from a newgroup (SFB).*

```
Identify/model potential confusion pairs
Build significant DGs for them
  Ensure sufficient instances of pair
  Collect legal eigenunit environments
  Analyze contexts of size one to limit
    if not significant data abort
    if useful store and continue
Scan and correct text sample
  For each potential confused word
    Collect maximal eigencontexts
    Report biggest context above threshold
```

Figure 1: *Summary of DG methodology used.*

the statistics for this pair invalidated our assumption of ergodicity across the three different 12000 line test corpora used. Also our initial prototype could not distinguish 'a/an' correctly. Conversely, on supposedly correct published text, we found six errors which had been missed by human proofreaders and commercial systems alike. For the record, these errors consisted of four 'was/were', one 'affect/effect', and one 'are/our' substitution. The first two errors are clear syntactic errors where the semantics is essentially the same. The second is a very common phono-frequens where, as different parts of speech, resolution is again straightforward. Note that a 95% precision setting should have been sufficient to find them, but would have eliminated around 80% of the false errors. The most difficult of these errors to resolve is the 'was/were' error because of the higher likelihood of a parenthetic intervention, which also contributes to the problem with 'its/it's'.

An example from (THC) demonstrating an unlikely usage of 'its', which requires a context of more than ten words to resolve, illustrates the problem of parenthetic intervention:

```
Its specialty magazines, such
as *Telephony,* *AT&T Technical
Journal,* *Telephone Engineer and
Management,* are decades old; they
make computer publications like
*Macworld* and *PC Week* look like
amateur johnny-come-latelies.
```

Another factor which causes severe problems with 'its/it's' is the extreme sensitivity of its differential grammar to contexts. Even the raw counts illustrate this quite clearly, and a far more representative training corpus will be needed to resolve the question of whether an adequate differential grammar can be built for this case: see Table 2.

The method we used to cope with the 'a/an' pair is simple and effective, but increases the number of additional affix classes from 13 to 26 as each is split according to whether it starts with a vowel or not. This increase the size of the eigenset to 174, but in addition we added 20 h-words and 2 y-words which take 'an', giving a total of 196 eigenunits. We illustrate what the eigenset now looks like in Table 3, where we present the top 15 eigenunits and their occurrence counts.

The affix information in Table 3 is equivalent to the cross-product of 26 prefixes with 13 suffixes (counting the 0-morph) and would have tripled the number of classes required if we hadn't made the preclassification into consonant and vowel. This is relevant as we go on to consider how our affix information could be derived automatically.

One of stated our aims was to seek to learn the syntactic information we use, but, in fact, we have used a set of 12 hand-chosen syntactically significant suffixes in the grammar checking discussed above, along with 150 words chosen on the basis of frequency, to which we have now added a pair of phonologically motivated features. We have therefore experimented with the automated discovery of an appropriate set of words and affixes.

For this purpose we sought to derive a set of maximal Ngrams which were significant but were not part of any larger significant Ngrams. Allowing Ngrams of different sizes means we are double counting some strings, and it is thus usual to deduct from a given Ngram prefix the frequencies of all N+1-grams which it prefixes, and similarly for suffixes. Using these as significance measures, however, tends to lead to us picking up not only frequent words and affixes, but frequent phrases and all proper substrings of each of these. Furthermore the last character of a suffix may well be involved in many other words and suffixes and thus tends to appear more significant.

WORD	RSV	RSV-i	THC	THC-i	SFJ	SFJ-i
its	1344	1370	179	186	210	215
it's	0	0	49	113	1138	1644
TOTAL:	757523	757523	106433	106433	414114	414114

Table 2: *Corpus sensitivity of 'its/it's' shown with both case sensitive and insensitive (-i) counts taken respectively from the corpora RSV, THC and SFJ.*

PERCENT	COUNT	NGRAM	PERCENT	COUNT	NGRAM	PERCENT	COUNT	NGRAM
29	223225	C-	3	26854	C-s	1	9577	you
8	61033	the	2	21046	to	1	9140	for
5	40812	and	1	13398	C-ed	1	9079	a
4	37855	V-	1	11762	in	1	8009	i
4	31491	of	1	9874	he	1	7837	his

Table 3: *Most significant 15 eigenunits with frequencies. 'V-/C-' are respectively all words that a vowel/consonant that are not matched as words or with specific suffixes like 'C-s C-ed'. The corpus (RSV) was selected to be topically focussed and of convenient size (757523 words).*

We therefore used a related heuristic in which we required that a unit be significant in *both* contexts in order to be treated as significant, and achieved this by double discounting – subtracting counts for both prefixes and suffixes. Although this method was intended as only a rough ranking for examining the results, it did indeed provide more useful information than either of the more principled discounts or their maximum or sum, for which again frequent words were represented multiply. With our double discount, words which are almost always used as part of a bigger significant string will end up heavily negatively weighted, and thus the heuristic is likely to prefer to embed it in a larger string – see Table 4.

N	RIGHT	LEFT	BI	COUNT	NGRAM
6	2241	2241	2241	2241	#this#
6	2050	2050	2050	2050	#the#b
6	1907	1907	1907	1907	e#and#
5	1886	1886	1886	1886	#of#a
4	1840	1840	1840	1840	#me#
5	1809	2377	1809	2377	#had#
1	2233	2699	1801	3131	-
3	1767	2201	1767	2201	#ba
5	1761	2285	1761	2285	#one#
7	2228	3257	1734	3751	#their#
7	2074	1624	1624	2074	e,#and#
4	1597	1597	1597	1597	#we#
9	1591	2175	1591	2175	s#of#the#
3	1588	1588	1588	1588	's#
10	1579	1579	1579	1579	#from#the#

Table 4: *'#' represents space. These are the 15 most significant maximal strings from an experiment which sought to discover the eigenunits of RSV as discounted Ngrams. Significance threshold was set at the 99.99% level, and contexts were discounted by the frequency of any significant contexts which extended them to the right, to the left, or either.*

While our eigenwords and hand-selected suffixes tended to be proposed relatively quickly, it will be observed that many actually occurred as part of strings which crossed word boundaries. Moreover, without some segmentation information the technique is sensitive to the significance threshold, which has a direct influence on the length of the Ngrams proposed. Limiting to maximal space-bounded 'words' is however reasonable in this application, but since we need to include punctuation and numbers in our eigenclass, we do not to filter these out. The top 75 candidates then consist almost entirely of Unix eigenwords, plus corpus eigenwords 'god' and 'lord', some punctuation, some standard affixes, some combinations of punctuation and affixes, and some unexpected candidate affixes. In fact some of these candidates, '-e -es', are not at all unreasonable: '-es' is a variant of '-s' and both can fit in the same slot as '-ed'. But others, 'bo- ba- ne-', are harder to make sense of. The next 75 strings are similar with a higher proportion of affixes, both syntactic (6/12 now covered) and non-syntactic (20), as well as two unclassifiable sequences ('rai ob').

Thus, it is clearly easy to obtain a fair approximation to our list of eigenunits, and the fact that 10 or 20% of them may not satisfy our syntactic expectations does not preclude them from being useful and will not necessarily worsen the results. For example, we note that our 24 prefixes handle resp. 33% and 20% of the 'a/an' cases covered by our 'V*' and 'C*' classes. As long as we are not overwhelmed by poor candidates, our eigenset will still be able to meet its goal.

An automatically generated eigenset, of the same size as our original 172 eigenunit version, included 80 of our original eigenwords which covered 54% (the Unix 150 covers 60%) of the corpus, and included 7 of our original suffixes covering an additional 12% (our handpicked 12 cover 13%). On the other hand, it proved that one of our hand-selected suffixes was not very significant in the corpus ('-ic') and occurred only 92 times (the .0001 threshold sets significance at 75 occurences). The last of the other suffixes ('-ble') to be proposed had rank 503, again because of larger significant contexts '-able ble-', which caused it to be discounted as a suffix in its own right.

Forcing a word-boundary between words and punctuation increases the rate at which eigenunits are found, as combinations of letters and punctations constitute the majority of the dross. Word-internal apostrophe (but not hyphen) is treated as a letter for this purpose.

10 Conclusions

Differential Grammars allow high-order Ngram statistics to be focussed on the problem of deciding between the correct and one or more incorrect tokens, reducing Ngram contexts to environments based on high frequency eigentokens: words, numbers, punctuation and affixes. Using the 150 Unix eigenwords gives us a 50% likelihood that we will have a hit in any slot, while our 12 non-zero suffixes increases the coverage to 25%, ensuring that good syntactic relevance is obtained.

In further smaller experiments, we demonstrated that the 'a/an' distinction could be handled by splitting our suffix and open eigenunits into vowel and consonant subclasses. We further demonstrated that similarly appropriate eigenunits could be automatically derived on a discounted frequency basis, using a crude heuristic to order the potential eigenunits, while restricting them to the form of lexical, space-bounded, words. Experiments involving training with an automatically derived eigenset have yet to be performed, and will focus on deciding the optimum size of eigenset and development of an improved heuristic.

The eigenset has two functions: to allow us to reduce the size of the tree for a given performance level, and to allow us to reduce the role of genre and semantic related fluctuations in word frequencies by concentrating on features of relatively high syntactic significance. Increasing the size of the eigenset is expected to decrease performance due to increased noise after a certain point. Similarly, increasing the size of the eigenset may eventually tend to increase the size of the stored differential grammars without significant gain in precision.

The use of a significance factor in the training stage allowed the size of the trees generated by the differential grammar generator to be limited to what was necessary to achieve that level of precision on the training corpus, whilst the likelihood values stored in the tree allowed the user to be informed of the likelihood of an error (using colour or upon query), and to control the threshold for which errors would be reported.

Maximum diameter is another parameter of the training stage, and experiments on optimal size and the role of diameter in relation to syntactic and semantic words were undertaken early on in setting 10 as the size beyond which environments were unlikely to reach significance. If generation of the grammar was stopped due to lack of significance, the problem was often lack of data. If the search was terminated at maximum diameter it was an indication that the words were functionally similar, and most likely the same part of speech.

The differential grammar approach has proven to be a successful way of applying statistical, Ngram-like, techniques for practical grammar-checking in a modest computing environment, with useful grammar trees requiring of the order of 100 to 1000 bytes of storage per confused word pair in most cases. This report has concentrated on presenting empirical results for a single system, rather than on optimization of the system, and there remains considerable scope for investigation of the role of the system parameters and optimizing the eigenset, for which only the primary considerations have been outlined. The primary deficiency of the system is its inability to cope with arbitrarily long parentheses or subclauses which separate syntactically bound elements, but it is also rather sensitive the genre and representativeness of the training corpus.

Acknowledgements

This work was undertaken jointly with Philip Kernick who, in particular, implemented the user interfaces and carried out the experiments whose results are reported in Table 1 as part of an Honours project (Kernick, 1996).

References

Eugene Charniak. 1993. Statistical Language Learning. MIT Press.

Kenneth W. Church and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54.

Kenneth W. Church and William A. Gale. 1995. Poisson mixtures. *Journal of Natural Language Engineering*, 2:163–190.

Jim Entwisle and M. Groves. 1994. A method of parsing English based on sentence form. *NeMLaP*.

Steven P. Finch. 1993. Finding Structure in Language. Ph.D. Thesis, University of Edinburgh.

Ill editors. 1983. Special issue on Word-Sense Disambiguation. *Computational Linguistics* 9#3-4.

E. Johnson. 1992 The Ideal Grammar and Style Checker Text Technology 2#4

Philip Kernick. 1996. A Statistical Grammar Checker. Honours Thesis, Flinders University of South Australia

Adam Kilgarriff 1996 Which words are particularly characteristic of a text? A survey of statistical approaches. ITRI Technical Report, University of Brighton

Donald E. Knuth. 1973. *The Art of Computer Programming*, Vol.3. Addison Wesley.

RSV: United Bible Society. The Bible. Revised Standard Version.

SFB: News Contributors. 1996. Contributions to usenet group aus.sf.babylon5.

SFJ: J. M. Straczynski. 1997. Contributions to usenet group alt.tv.babylon-5, Jul 93 to Feb 97.

SFK: P. Kernick. 1996. Science Fiction writings

THC: Bruce Sterling. 1992. *The Hacker Crackdown.* Bantam Books. Electronic version 1994.

TIPSTER Information Retrieval. 1994. Text Research Collection Vol.2. *University of Pennsylvania*

B. E. Wampler. 1995. Risks of grammar checkers. *The Risks Digest* 17#54, Dec 1995.

Patrick Winston. 1993. *Artificial Intelligence. 3rd Edition.* Addison Wesley.