All-word prediction as the ultimate confusable disambiguation

Antal van den Bosch
ILK / Dept. of Language and Information Science, Tilburg University
P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands
Antal.vdnBosch@uvt.nl

Abstract

We present a classification-based word prediction model based on IGTREE, a decision-tree induction algorithm with favorable scaling abilities and a functional equivalence to n-gram models with back-off smoothing. Through a first series of experiments, in which we train on Reuters newswire text and test either on the same type of data or on general or fictional text, we demonstrate that the system exhibits log-linear increases in prediction accuracy with increasing numbers of training examples. Trained on 30 million words of newswire text, prediction accuracies range between 12.6% on fictional text and 42.2% on newswire text. In a second series of experiments we compare all-words prediction with confusable prediction, i.e., the same task, but specialized to predicting among limited sets of words. Confusable prediction yields high accuracies on nine example confusable sets in all genres of text. The confusable approach outperforms the all-words-prediction approach, but with more data the difference decreases.

1 Introduction

Word prediction is an intriguing language engineering semi-product. Arguably it is the "archetypical prediction problem in natural language processing" (Even-Zohar and Roth, 2000). It is usually not an engineering end in itself to predict the next word in a sequence, or fill in a blanked-out word in a sequence. Yet, it could be an asset in higher-level proofing or authoring tools, e.g. to be able to automatically discern among confusables and thereby to detect confusable errors (Golding and Roth, 1999; Even-Zohar and Roth, 2000; Banko and Brill, 2001; Huang and Powers, 2001). It could alleviate problems with low-frequency and unknown words in natural language processing and information retrieval, by replacing them with likely and higher-frequency alternatives that carry similar information. And also, since the task of word prediction is a direct interpretation of language modeling, a word prediction system could provide useful information for to be used in speech recognition systems.

A unique aspect of the word prediction task, as compared to most other tasks in natural language processing, is that real-world examples abound in large amounts. Any digitized text can be used as training material for a word prediction system capable of learning from examples, and nowadays gigascale and terascale document collections are available for research purposes.

A specific type of word prediction is confusable prediction, i.e., learn to predict among limited sets of confusable words such as *to*/*two*/*too* and *there*/*their*/*they're* (Golding and Roth, 1999; Banko and Brill, 2001). Having trained a confusable predictor on occurrences of words within a confusable set, it can be applied to any new occurrence of a word from the set; if its prediction based on the context deviates from the word actually present, then

Workshop on Computationally Hard Problemsand Joint Inference in Speech and Language Processing, pages 25–32,
New York City, New York, June 2006. ©2006 Association for Computational Linguistics

this word might be a confusable error, and the classifier's prediction might be its correction. Confusable prediction and correction is a strong asset in proofing tools.

In this paper we generalize the word prediction task to predicting *any* word in context. This is basically the task of a generic language model. An explicit choice for the particular study on "all-words" prediction is to encode context only by words, and not by any higher-level linguistic non-terminals which have been investigated in related work on word prediction (Wu et al., 1999; Even-Zohar and Roth, 2000). This choice leaves open the question how the same tasks can be learned from examples when non-terminal symbols are taken into account as well.

The choice for our algorithm, a decision-tree approximation of k-nearest-neigbor (k-NN) based or memory-based learning, is motivated by the fact that, as we describe later in this paper, this particular algorithm can scale up to predicting tens of thousands of words, while simultaneously being able to scale up to tens of millions of examples as training material, predicting words at useful rates of hundreds to thousands of words per second. Another motivation for our choice is that our decision-tree approximation of k-nearest neighbor classification is functionally equivalent to back-off smoothing (Zavrel and Daelemans, 1997); not only does it share its performance capacities with n-gram models with back-off smoothing, it also shares its scaling abilities with these models, while being able to handle large values of n.

The article is structured as follows. In Section 2 we describe what data we selected for our experiments, and we provide an overview of the experimental methodology used throughout the experiments, including a description of the IGTREE algorithm central to our study. In Section 3 the results of the word prediction experiments are presented, and the subsequent Section 4 contains the experimental results of the experiments on confusables. We briefly relate our work to earlier work that inspired the current study in Section 5. The results are discussed, and conclusions are drawn in Section 6.

2 Data preparation and experimental setup

First, we identify the textual corpora used. We then describe the general experimental setup of learning curve experiments, and the IGTREE decision-tree induction algorithm used throughout all experiments.

2.1 Data

To generate our word prediction examples, we used the "Reuters Corpus Volume 1 (English Language, 1996-08-20 to 1997-08-19)"[1]. We tokenized this corpus with a rule-based tokenizer, and used all 130,396,703 word and punctuation tokens for experimentation. In the remainder of the article we make no difference between words and punctuation markers; both are regarded as tokens. We separated the final 100,000 tokens as a held-out test set, henceforth referred to as REUTERS, and kept the rest as training set, henceforth TRAIN-REUTERS.

Additionally, we selected two test sets taken from different corpora. First, we used the Project Gutenberg[2] version of the novel *Alice's Adventures in Wonderland* by Lewis Carroll (Carroll, 1865), henceforth ALICE. As the third test set we selected all tokens of the Brown corpus part of the Penn Treebank (Marcus et al., 1993), a selected portion of the original one-million word Brown corpus (Kučera and Francis, 1967), a collection of samples of American English in many different genres, from sources printed in 1961; we refer to this test set as BROWN. In sum, we have three test sets, covering texts from the same genre and source as the training data, a fictional novel, and a mix of genres wider than the training set.

Table 1 summarizes the key training and test set statistics. As the table shows, the cross-domain coverages for unigrams and bigrams are rather low; not only are these numbers the best-case performance ceilings, they also imply that a lot of contextual information used by the machine learning method used in this paper will be partly unknown to the learner, especially in texts from other domains than the training set.

[1]For availability of the Reuters corpus, see `http://about.reuters.com/researchandstandards/corpus/`.

[2]Project Gutenberg: `http://www.gutenberg.net`.

Data set	Genre	# Tokens	Coverage (%)	
TRAIN-REUTERS	news	30 million	unigram	bigram
REUTERS	news	100,000	91.0	83.6
ALICE	fiction	33,361	85.2	70.1
BROWN	mixed	453,446	75.9	72.3

Table 1: Training and test set sources, genres, sizes in terms of numbers of tokens, and unigram and bigram coverage (%) of the training set on the test sets.

2.2 Experimental setup

All experiments described in this article take the form of learning curve experiments (Banko and Brill, 2001), in which a sequence of training sets is generated with increasing size, where each size training set is used to train a model for word prediction, which is subsequently tested on a held-out test set – which is fixed throughout the whole learning curve experiment. Training set sizes are exponentially grown, as earlier studies have shown that at a linear scale, performance effects tend to decrease in size, but that when measured with exponentially growing training sets, near-constant (i.e. log-linear) improvements are observed (Banko and Brill, 2001).

We create incrementally-sized training sets for the word prediction task on the basis of the TRAIN-REUTERS set. Each training subset is created backward from the point at which the final 100,000-word REUTERS set starts. The increments are exponential with base number 10, and for every power of 10 we cut off training sets at n times that power, where $n = 1, 2, 3, \ldots, 8, 9$ (for example, $10, 20, \ldots, 80, 90$).

The actual examples to learn from are created by *windowing* over all sequences of tokens. We encode examples by taking a left context window spanning seven tokens, and a right context also spanning seven tokens. Thus, the task is represented by a growing number of examples, each characterized by 14 positional features carrying tokens as values, and one class label representing the word to be predicted. The choice for 14 is intended to cover at least the superficially most important positional features. We assume that a word more distant than seven positions left or right of a focus word will almost never be more informative for the task than any of the words within this scope.

2.3 IGTree

IGTree (Daelemans et al., 1997) is an algorithm for the top-down induction of decision trees. It compresses a database of labeled examples into a lossless-compression decision-tree structure that preserves the labeling information of all examples, and technically should be named a *trie* according to (Knuth, 1973). A labeled example is a feature-value vector, where features in our study represent a sequence of tokens representing context, associated with a symbolic class label representing the word to be predicted. An IGTREE is composed of nodes that each represent a partition of the original example database, and are labeled by the most frequent class of that partition. The root node of the trie thus represents the entire example database and carries the most frequent value as class label, while end nodes (leafs) represent a *homogeneous* partition of the database in which all examples have the same class label. A node is either a leaf, or is a non-ending node that branches out to nodes at a deeper level of the trie. Each branch represents a test on a feature value; branches fanning out of one node test on values of the same feature.

To attain high compression levels, IGTREE adopts the same heuristic that most other decision-tree induction algorithms adopt, such as C4.5 (Quinlan, 1993), which is to always branch out testing on the most informative, or most class-discriminative features first. Like C4.5, IGTREE uses information gain (IG) to estimate the most informative features. The IG of feature i is measured by computing the difference in uncertainty (i.e. entropy) between the situations without and with knowledge of the value of that feature with respect to predicting the class label: $IG_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v)$, where C is the set of class labels, V_i is the set of values for feature i, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. In contrast with C4.5, IGTREE computes the IG of all features once on the full database of training examples, makes a feature ordering once on these computed IG values, and uses this ordering throughout the whole trie.

Another difference with C4.5 is that IGTREE does not prune its produced trie, so that it performs a lossless compression of the labeling information of the original example database. As long as the database does not contain fully ambiguous examples (with the same features, but different class labels), the trie produced by IGTREE is able to reproduce the classifications of all examples in the original example database perfectly.

Due to the fact that IGTREE computes the IG of all features once, it is functionally equivalent to IB1-IG (Daelemans et al., 1999), a k-nearest neighbor classifier for symbolic features, with $k = 1$ and using a particular feature weighting in the similarity function in which the weight of each feature is larger than the sum of all weights of features with a lower weight (e.g. as in the exponential sequence $1, 2, 4, 8, \ldots$ where $2 > 1$, $4 > (1 + 2)$, $8 > (1 + 2 + 4)$, etc.). Both algorithms will base their classification on the example that matches on most features, ordered by their IG, and guess a majority class of the set of examples represented at the level of mismatching. IGTREE, therefore, can be seen as an approximation of IB1-IG with $k = 1$ that has favorable asymptotic complexities as compared to IB1-IG.

IGTREE's computational bottleneck is the trie construction process, which has an asymptotic complexity of $O(n\, lg(v)\, f)$ of CPU, where n is the number of training examples, v is the average branching factor of IGTREE (how many branches fan out of a node, on average), and f is the number of features. Storing the trie, on the other hand, costs $O(n)$ in memory, which is less than the $O(n\ f)$ of IB1-IG. Classification in IGTREE takes an efficient $O(f\ lg(v))$ of CPU, versus the cumbersome worst-case $O(n\ f)$ of IB1-IG, that is, in the typical case that n is much higher than f or v.

Interestingly, IGTREE is functionally equivalent to back-off smoothing (Zavrel and Daelemans, 1997), with the IG of the features determining the order in which to back off, which in the case of word prediction tends to be from the outer context to the inner context of the immediately neighboring words. Like with probabilistic n-gram based models with a back-off smoothing scheme, IGTREE will prefer matches that are as exact as possible (e.g. matching on all 14 features), but will back-off by disregarding lesser important features first, down to a simple bigram model drawing on the most important feature, the immediately preceding left word. In sum, IGTREE shares its scaling abilities with n-gram models, and its implementation allows it to handle large values of n.

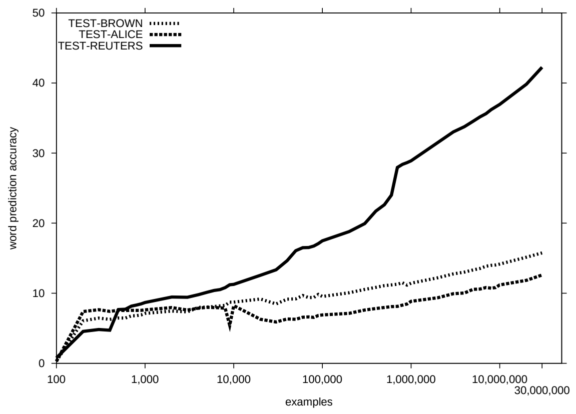

Figure 1: Learning curves of word prediction accuracies of IGTREE trained on TRAIN-REUTERS, and tested on REUTERS, ALICE, and BROWN.

3 All-words prediction

3.1 Learning curve experiments

The word prediction accuracy learning curves computed on the three test sets, and trained on increasing portions of TRAIN-REUTERS, are displayed in Figure 1. The best accuracy observed is 42.2% with 30 million training examples, on REUTERS. Apparently, training and testing on the same type of data yields markedly higher prediction accuracies than testing on a different-type corpus. Accuracies on BROWN are slightly higher than on ALICE, but the difference is small; at 30 million training examples, the accuracy on ALICE is 12.6%, and on BROWN 15.8%.

A second observation is that all three learning curves are progressing upward with more training examples, and roughly at a constant log-linear rate. When estimating the rates after about 50,000 examples (before which the curves appear to be more volatile), with every tenfold increase of the number of training examples the prediction accuracy on REUTERS increases by a constant rate of about 8%, while the increases on ALICE and BROWN are both about 2% at every tenfold.

3.2 Memory requirements and classification speed

The numbers of nodes exhibit an interesting sublinear relation with respect to the number of training examples, which is in line with the asymptotic complexity order $O(n)$, where n is the number of training instances. An increasingly sublinear amount of nodes is necessary; while at 10,000 training instances the number of nodes is 7,759 (0.77 nodes per instance), at 1 million instances the number of nodes is 652,252 (0.65 nodes per instance), and at 30 million instances the number of nodes is 15,956,878 (0.53 nodes per instance).

A factor in classification speed is the average amount of branching. Conceivably, the word prediction task can lead to a large branching factor, especially in the higher levels of the tree. However, not every word can be the neighbor of every other word in finite amounts of text. To estimate the average branching factor of a tree we compute the fth root of the total number of nodes (f being the number of features, i.e. 14). The largest decision tree currently constructed is the one on the basis of a training set of 30 million examples, having 15,956,878 nodes. This tree has an average branching factor of $\sqrt[14]{15,956,878} \approx 3.27$; all other trees have smaller branching factors. Together with the fact that we have but 14 features, and the asymptotic complexity order of classification is $O(f\ lg(v))$, where v is the average branching factor, classification can be expected to be fast. Indeed, depending on the machine's CPU on which the experiment is run, we observe quite favorable classification speeds. Figure 2 displays the various speeds (in terms of the number of test tokens predicted per second) attained on the three test sets[3]. The best prediction accuracies are still attained at classification speeds of over a hundred predicted tokens per second. Two other relevant observations are that first, the classification speed hardly differs between the three test sets (BROWN is classified only slightly slower than the other two test sets), indicating that the classifier is spending a roughly comparable amount of searching through the decision trees regardless of genre differences. Second, the decrease in speed settles

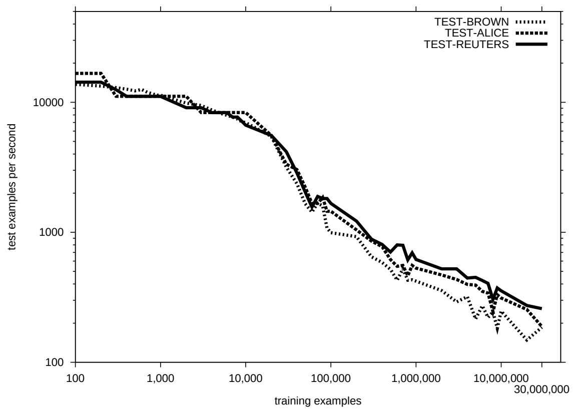

Figure 2: Word prediction speed, in terms of the number of classified test examples per second, measured on the three test sets, with increasing training examples. Both axes have a logarithmic scale.

on a low log-linear rate after about one million examples. Thus, while trees grow linearly, and accuracy increases log-linearly, the speed of classification slowly diminishes at decreasing rates.

4 Confusables

Word prediction from context can be considered a very hard task, due to the many choices open to the predictor at many points in the sequence. Predicting content words, for example, is often only possible through subtle contextual clues or by having the appropriate domain or world knowledge, or intimate knowledge of the writer's social context and intentions. In contrast, certain function words tend to be predictable due to the positions they take in syntactic phrase structure; their high frequency tends to ensure that plenty of examples of them in context are available.

Due to the important role of function words in syntactic structure, it can be quite disruptive for a parser and for human readers alike to encounter a mistyped function word that in its intended form is another function word. In fact, confusable errors between frequent forms occur relatively frequently. Examples of these so-called confusables in English are *there* versus *their* and the contraction *they're*; or the duo *than* and *then*. Confusables can arise from having the same pronunciation (homophones), or having very similar pronunciation (*country* or *county*) or spelling (*dessert*, *desert*), hav-

[3]Measurements were made on a GNU/Linux x86-based machine with 2.0 Ghz AMD Opteron processors.

ing very close lexical semantics (as between *among* and *between*), or being inflections or case variants of the same stem (*I* versus *me*, or *walk* versus *walks*), and may stem from a lack of concentration or experience by the writer.

Distinguishing between confusables is essentially the same task as word prediction, except that the number of alternative outcomes is small, e.g. two or three, rather than thousands or more. The typical application setting is also more specific: given that a writer has produced a text (e.g. a sentence in a word processor), it is possible to check the correctness of each occurrence of a word known to be part of a pair or triple of confusables.

We performed a series of experiments on disambiguating nine frequent confusables in English adopted from (Golding and Roth, 1999). We employed an experimental setting in which we use the same experimental data as before, in which only examples of the confusable words are drawn – note that we ignore possible confusable errors in both training and test set. This data set generation procedure reduces the amount of examples considerably. Despite having over 130 million words in TRAIN-REUTERS, frequent words such as *there* and *than* occur just over 100,000 times. To be able to run learning curves with more than this relatively small amount of examples, we expanded our training material with the New York Times of 1994 to 2002 (henceforth TRAIN-NYT), part of the English Gigaword collection published by the Linguistic Data Consortium, offering 1,096,950,281 tokens.

As a first illustration of the experimental outcomes, we focus on the three-way confusable *there* – *their* – *they're* for which we trained one classifier, which we henceforth refer to as a confusable expert. The learning curve results of this confusable expert are displayed in Figure 3 as the top three graphs. The logarithmic x-axis displays the full number of instances from TRAIN-REUTERS up to 130.3 million examples, and from TRAIN-NYT after this point. Counter to the learning curves in the all-words prediction experiments, and to the observation by (Banko and Brill, 2001), the learning curves of this confusable triple in the three different data sets flatten, and converge, remarkably, to a roughly similar score of about 98%. The convergence only occurs after examples from TRAIN-NYT are added.

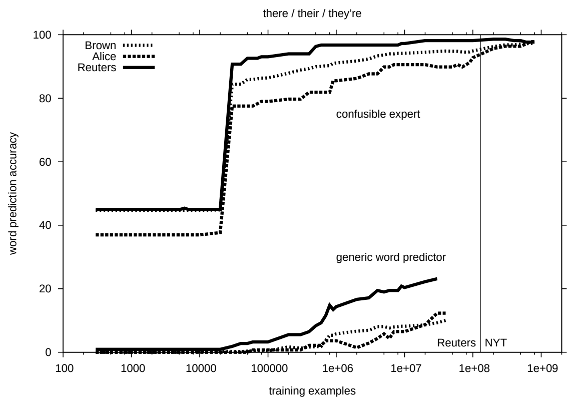

Figure 3: Learning curves in terms of word prediction accuracy on deciding between the confusable pair *there*, *their*, and *they're*, by IGTREE trained on TRAIN-REUTERS, and tested on REUTERS, ALICE, and BROWN. The top graphs are accuracies attained by the confusable expert; the bottom graphs are attained by the all-words predictor trained on TRAIN-REUTERS until 130 million examples, and on TRAIN-NYT beyond (marked by the vertical bar).

In the bottom of the same Figure 3 we have also plotted the word prediction accuracies on the three words *there*, *their*, and *they're* attained by the all-words predictor described in the previous section on the three test sets. The accuracies, or rather recall figures (i.e. the percentage of occurrences of the three words in the test sets which are correctly predicted as such), are considerably lower than those on the confusable disambiguation task.

Table 2 presents the experimental results obtained on nine confusable sets when training and testing on Reuters material. The third column lists the accuracy (or recall) scores of the all-words word prediction system at the maximal training set size of 30 million labeled examples. The fourth columns lists the accuracies attained by the confusable expert for the particular confusable pair or triple, measured at 30 million training examples, from which each particular confusable expert's examples are extracted. The amount of examples varies for the selected confusable sets, as can be seen in the second column.

Scores attained by the all-words predictor on these words vary from below 10% for relatively low-frequent words to around 60% for the more frequent confusables; the latter numbers are higher than the

Confusable set	Number of examples	Accuracy (%) by all-words prediction	confus. expert
cite - site - sight	2,286	0.0	100.0
accept - except	3,833	46.2	76.9
affect - effect	4,640	7.7	87.9
fewer - less	6,503	4.7	95.2
among - between	27,025	18.9	96.7
I - me	28,835	55.9	98.0
than - then	31,478	59.4	97.2
there - their - they're	58,081	23.1	96.8
to - too - two	553,453	60.6	93.4

Table 2: Disambiguation scores on nine confusable set, attained by the all-words prediction classifier trained on 30 million examples of TRAIN-REUTERS, and by confusable experts on the same training set. The second column displays the number of examples of each confusable set in the 30-million word training set; the list is ordered on this column.

	Accuracy on test set (%)		
Confusable set	REUTERS	ALICE	BROWN
cite - site - sight	100.0	100.0	69.0
accept - except	84.6	100.0	97.0
affect - effect	92.3	100.0	89.5
fewer - less	90.5	100.0	97.2
among - between	94.4	77.8	74.4
I - me	99.0	98.3	98.3
than - then	97.2	92.9	95.8
there - their - they're	98.1	97.8	97.3
to - too - two	94.3	93.4	92.9

Table 3: Disambiguation scores on nine confusable set, attained by confusable experts trained on examples extracted from 1 billion words of text from TRAIN-REUTERS plus TRAIN-NYT, on the three test sets.

overall accuracy of this system on REUTERS. Nevertheless they are considerably lower than the scores attained by the confusable disambiguation classifiers, while being trained on many more examples (i.e., all 30 million available). Most of the confusable disambiguation classifiers attain accuracies of well above 90%.

When the learning curves are continued beyond TRAIN-REUTERS into TRAIN-NYT, about a thousand times as many training examples can be gathered as training data for the confusable experts. Table 3 displays the nine confusable expert's scores after being trained on examples extracted from a total of one billion words of text, measured on all three test sets. Apart from a few outliers, most scores are above 90%, and more importantly, the scores on ALICE and BROWN do not seriously lag behind those on REUTERS; some are even better.

5 Related work

As remarked in the cases reported in the literature directly related to the current article, word prediction is a core task to natural language processing, and one of the few that takes no annotation layer to provide data for supervised machine learning and probabilistic modeling (Golding and Roth, 1999; Even-Zohar and Roth, 2000; Banko and Brill, 2001). Our discrete, classificatio-nased approach has the same goal as probabilistic methods for language modeling for automatic speech recognition (Jelinek, 1998), and is also functionally equivalent to n-gram models with back-off smoothing (Zavrel and Daelemans, 1997).

The papers by Golding and Roth, and Banko and Brill on confusable correction focus on the more common type of *than*/*then* confusion that occurs a lot in the process of text production. Both pairs of authors use the confusable correction task to illustrate scaling issues, as we have. Golding and Roth illustrate that multiplicative weight-updating algorithms such as Winnow can deal with immense input feature spaces, where for each single classification only a small number of features is actually relevant (Golding and Roth, 1999). With IGTREE we have an arguably competitive efficient, but one-shot learning algorithm; IGTREE does not need an iterative procedure to set weights, and can also handle a large feature space. Instead of viewing all positional features as containers of thousands of atomic word features, it treats the positional features as the basic tests, branching on the word values in the tree.

More generally, as a precursor to the above-mentioned work, confusable disambiguation has been investigated in a string of papers discussing the application of various machine learning algorithms to the task (Yarowsky, 1994; Golding, 1995; Mangu and Brill, 1997; Huang and Powers, 2001).

6 Discussion

In this article we explored the scaling abilities of IGTREE, a simple decision-tree algorithm with favorable asymptotic complexities with respect to multi-label classification tasks. IGTREE is applied to word prediction, a task for which virtually unlimited amounts of training examples are available, with very large amounts of predictable class labels; and confusable disambiguation, a specialization of word prediction focusing on small sets of confusable words. Best results are 42.2% correctly predicted tokens (words and punctuation markers) when training and testing on data from the *Reuters* newswire corpus; and confusable disambiguation accuracies of well above 90%. Memory requirements and speeds were shown to be realistic.

Analysing the results of the learning curve experiments with increasing amounts of training examples, we observe that better word prediction accuracy can be attained simply by adding more training examples, and that the progress in accuracy proceeds at a log-linear rate. The best rate we observed was an 8% increase in performance every tenfold multiplication of the number of training examples, when training and testing on the same data.

Despite the fact that all-words prediction lags behind in disambiguating confusibles, in comparison with classifiers that are focused on disambiguating single sets of confusibles, we see that this lag is only relative to the amount of training material available.

Acknowledgements

This research was funded by the Netherlands Organisation for Scientific Research (NWO). The author wishes to thank Ko van der Sloot for programming assistance.

References

M. Banko and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics.

L. Carroll. 1865. *Alice's Adventures in Wonderland*. Project Gutenberg.

W. Daelemans, A. Van den Bosch, and A. Weijters. 1997. IGTree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.

W. Daelemans, A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.

Y. Even-Zohar and D. Roth. 2000. A classification approach to word prediction. In *Proceedings of the First North-American Conference on Computational Linguistics*, pages 124–131, New Brunswick, NJ. ACL.

A.R. Golding and D. Roth. 1999. A Winnow-Based Approach to Context-Sensitive Spelling Correction. *Machine Learning*, 34(1–3):107–130.

A. R. Golding. 1995. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the 3rd workshop on very large corpora, ACL-95*.

J. H. Huang and D. W. Powers. 2001. Large scale experiments on correction of confused words. In *Australasian Computer Science Conference Proceedings*, pages 77–82, Queensland AU. Bond University.

F. Jelinek. 1998. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA.

D. E. Knuth. 1973. *The art of computer programming*, volume 3: Sorting and searching. Addison-Wesley, Reading, MA.

H. Kučera and W. N. Francis. 1967. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

L. Mangu and E. Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the International Conference on Machine Learning*, pages 187–194.

M. Marcus, S. Santorini, and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

J.R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

D. Wu, Z. Sui, and J. Zhao. 1999. An information-based method for selecting feature types for word prediction. In *Proceedings of the Sixth European Conference on Speech Communication and Technology, EUROSPEECH'99*, Budapest.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French. In *Proceedings of the Annual Meeting of the ACL*, pages 88–95.

J. Zavrel and W. Daelemans. 1997. Memory-based learning: Using similarity for smoothing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 436–443.