Scalable classification-based word prediction and confusible correction

Antal van den Bosch

ILK / 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. Through a first series of experiments we demonstrate that the system exhibits log-linear increases in prediction accuracy and decreases in* discrete perplexity, *a new evaluation metric, with increasing numbers of training examples. The induced trees grow linearly with the amount of training examples. Trained on 30 million words of newswire text, prediction accuracies reach 42.2% on the same type of text. In a second series of experiments we show that this generic approach to word prediction can be specialized to confusible prediction, yielding high accuracies on nine example confusible sets in all genres of text. The confusible-specific approach outperforms the generic word-prediction approach, but with more data the difference decreases.*

RÉSUMÉ. *Cet article présente un modèle pour la tâche de prédiction de mots, considérée ici comme une tâche de classification. Ce modèle repose sur l'utilisation de* IGTREE, *un algorithme d'inférence d'arbre de décision capable de traiter à la fois un grand nombre de classes et d'exemples d'apprentissage. À travers une première série d'expérimentations nous montrons que la capacité de prédiction du modèle augmente log-linéairement avec le nombre d'exemples d'entraînement. Le même comportement est obtenu avec la* perplexité discrète, *une nouvelle métrique introduite pour la tâche de prédiction de mots ; la taille des arbres inférés croît, elle, linéairement. Lorsque notre modèle est entraîné sur un corpus journalistique de 30 millions de mots, le nombre de mots correctement prédits est de 42.2 % sur des textes journalistiques. Une seconde série d'expérimentations démontre que ce prédicteur générique peut être spécialisé pour traiter des configurations dans lesquelles l'ensemble des mots à prédire se restreint à un petit ensemble. Le modèle spécialisé atteint des meilleurs résultats que le classifieur générique.*

KEYWORDS: *word prediction, language modeling, induction of decision trees, perplexity.*

MOTS-CLÉS : *prédiction de mots, modèles de langage, inférence d'arbres de décision, perplexité.*

1. Introduction

Word prediction is an intriguing language engineering semi-product. Arguably it is the "archetypal 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 is an asset in higher-level proofing or authoring tools, e.g. to be able to automatically discern among confusibles and thereby to detect confusible errors (Golding and Roth, 1999; Even-Zohar and Roth, 2000; Banko and Brill, 2001; Huang and Powers, 2001), or to suggest words in a word processing environment, both under normal circumstances and in special cases such as language learning or augmentative communication (Wood, 1996; Garay-Vitoria and González-Abascal, 1997). 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; it could provide answers to some questions in question answering systems by filling in blanks. And finally, since the task of word prediction is a direct interpretation of language modeling, a word prediction system could provide useful information for language modeling components in speech recognition systems (Wu *et al.*, 1999) and information retrieval systems (Hiemstra, 2001).

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, without any annotation cost involved and hardly without data collection hurdles. Nowadays, gigascale and terascale document collections[1] are available for research purposes. Any digitized text can be used as training material for a word prediction system capable of learning from examples, as employed in the study described in this article. Banko and Brill (2001) use the abundance aspect of the task to perform learning curve experiments on confusible disambiguation, extracting their training material from English text corpora containing up to one billion words.

More general than focusing on limited numbers of confusible sets such as *to*/*two*/*too* and *there*/*their*/*they're* (Golding and Roth, 1999; Banko and Brill, 2001) we define the word prediction task as to predict *any* word in context, where we assume we have a left context of words, or both left and right contexts, which are both realistic variants of this task. In the first, in which only the left context of a word is given, the word predictor is relying only on the recent history to suggest continuations to the user, or to support a directional language model for speech recognition (Jelinek, 1998). In the second variant, both left and right contexts are given, and the word in the middle is queried. For example, given a left context of *Alice was beginning to get very* and a right context of *of sitting by her sister on the bank*, the system would ideally predict that the word in the middle is *tired*. A realistic use of this prediction based on context on both sides is in spelling correction, where a given but possibly

1. Two examples of large-scale corpora are the data used in the Terabyte Tracks of TREC 2004 and 2005, http://www-nlpir.nist.gov/projects/terabyte/, and the English Gigaword corpus produced by the Linguistic Data Consortium, at http://www.ldc.upenn.edu/

misspelled word may be blanked out and the word predictor may guess what the most appropriate form is given the left and right contexts (Reynaert, 2004; Reynaert, 2005). In this article we largely focus on the task of having both left and right context, but we also provide results on the left-context-only task.

It is stating the obvious that the word prediction task in both variants is hard, and surely impossible to perform perfectly. In the example of the aforementioned first sentence of *Alice's Adventures in Wonderland* (Carroll, 1865), the missing word *tired* could have been a host of other likely and unlikely words; machines nor humans can predict which. No word prediction method will ever produce 100% accurate predictions on any text; the upper bound will typically be much lower. We expect, however, to do relatively well on predicting function words. The problem of word prediction in English and similar languages could be seen as a combination of two problems: predicting function words and content words. Predicting function words involves being able to recognize syntactic structure and lexical preferences of neighboring content words. Predicting content words involves, ultimately, knowing what a sentence and the text around it is about and how it relates to the world and the social context which it was intended for.

Function words abound, while content words are less frequent to rare (Zipf, 1935). Any digital text will provide many examples of the same function words in context, but may provide only few or single examples of rare content words in context at the same time. A machine learning algorithm will tend to struggle with generalizing from these rare examples to predict the same word again, unless the context is quite or exactly the same as during training – provided that the learning algorithm has remembered the exact context.

In this study we 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 is motivated by earlier findings that with more data, there are increasingly less benefits from non-terminal levels such as part-of-speech tags in natural language processing tasks (Van den Bosch and Buchholz, 2002) — but it certainly leaves open the question how the same tasks can be learned from examples when non-terminal symbols are taken into account as well.

From an engineering point of view, our study addresses two new questions that have only partially been investigated in earlier related work:

1) How will a machine learning system applied to word prediction scale up to tens of thousands of **classes** or more, rather than just the two or three classes of a confusible set? Such a system will need to choose among the tens of thousands of words it has seen in the training material, and this might not be practically feasible given that for some machine learning algorithms the number of classes is a factor that adds complexity to the costs of storing the learned model or in classification (e.g., as with binary support-vector machines).

2) How will this learner scale up to being presented with millions of **examples** of

words in context as training material, and how will it scale up in classification mode, making the predictions? The typical experiment in which machine learning algorithms are applied to natural language processing tasks is based on hundreds to thousands to maximally a few million examples, while for this task many millions of examples can be easily generated. Also, some machine learning classifiers may be prohibitively slow in classifying even with moderate amounts of training material (e.g., with classifiers based on the k-nearest neighbor classification rule).

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 in this article, this particular algorithm has interesting scaling abilities in these two dimensions: it can scale up to predicting tens of thousands of words, while simultaneously scaling up to nearly ten million examples as training material, predicting words at useful rates of hundreds to thousands of words per second.

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, as well as a proposal for *discrete perplexity* as an alternative evaluation metric besides word prediction accuracy. 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 confusibles. 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

In this section we outline the prerequisites for the experiments described in the subsequent sections. 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. To conclude the section, we list the four evaluation metrics used, including the new *discrete perplexity* metric.

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)”[2]. 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 TEST-REUTERS, and kept the rest as training set, henceforth TRAIN-REUTERS.

2. For availability of the Reuters corpus, see http://about.reuters.com/researchandstandards/corpus/

Data set	Source	Genre	Number of tokens
TRAIN-REUTERS	Reuters Corpus Volume 1	newswire	130,396,703
TEST-REUTERS	Reuters Corpus Volume 1	newswire	100,000
TEST-ALICE	Alice in Wonderland	fiction	33,361
TEST-BROWN	Brown (Penn Treebank)	mixed	453,446

Table 1. *Training and test set sources, genres, and sizes in terms of numbers of tokens (words plus punctuation marks)*

Additionally, we selected two test sets taken from different corpora. First, we used the Project Gutenberg[3] version of the novel *Alice's Adventures in Wonderland* (also known as *Alice in Wonderland*) by Lewis Carroll (Carroll, 1865), henceforth TEST-ALICE. As the third test set we selected all tokens of the Brown corpus part of the Penn Treebank (Marcus *et al.*, 1993). This is 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 TEST-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.

2.2. *Experimental setup*

All experiments described in this article take the form of learning curve experiments (Banko and Brill, 2001; Van den Bosch and Buchholz, 2002). In a learning curve experiment a sequence of training sets is generated with increasing size. Each size training set is used to train a model for word prediction, which is subsequently tested on a held-out test set that 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; Van den Bosch and Buchholz, 2002).

We create incrementally-sized training sets for the word prediction task on the basis of the 130,296,703-token TRAIN-REUTERS set. Each training subset is created backward from the point at which the final 100,000-word TEST-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,

3. Project Gutenberg: http://www.gutenberg.net

Windowed input context		Word to be
Left context	Right context	predicted
once or twice she had peeped into	book her sister was reading , but	the
or twice she had peeped into the	her sister was reading , but it	book
twice she had peeped into the book	sister was reading , but it had	her
she had peeped into the book her	was reading , but it had no	sister
had peeped into the book her sister	reading , but it had no pictures	was

Table 2. *Five windowed examples created from the first sentence of* Alice's Adventures in Wonderland*: seven left and right neighboring tokens mapping to the middle word to be predicted*

and a right context also spanning seven tokens. (Later we also report on experiments in which examples only contain a left context of seven words.) Thus, the task is represented by a variable 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: the immediately neighboring tokens. We assume that a word more distant than seven positions left or right of a focus word will almost never be informative for the task. Although a strong assumption, it is supported by empirical evidence that most collocational word pairs are positioned less than five words apart (Martin *et al.*, 1983). Table 2 displays five examples created for the TEST-ALICE test set from *Alice's Adventures in Wonderland*.

As a variant of this basic setup, we employ simple frequency-of-occurrence-based *muting*, as a brute heuristic for dealing with low-frequency words. The muting heuristic deletes all examples of words in the training material that occur below a certain user-defined occurrence threshold. When occurring as context tokens in another word's example, words below the threshold are converted to a special symbol. We varied in all these experiments of training and testing on *Reuters* data the frequency threshold for muting, from 1 (no muting) to 10, 100, and 1000. Muting the training set effectively focuses the word predictor to higher-frequent words. We do not mute test material – the underlying reasoning is that the word predictor should in all cases be evaluated on an "all-words" prediction accuracy. Obviously, no test words muted in the training set will be predicted, but on the other hand muting could save the classifier from storing low-frequency examples that do not reoccur in the test data, and gives the classifier a statistically more reliable number of examples per word it is trained to predict.

2.3. *IGTree*

IGTree[4] (Daelemans, Van den Bosch, and Weijters, 1997a) 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 represent some input (here, a sequence of tokens representing context), associated with a symbolic class label representing some output (here, the word to be predicted). After the construction of the tree, it can be used to classify new examples not in the original database. A typical trie is composed of nodes that each represent a partition of the original example database, and 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), and CART (Breiman *et al.*, 1984), which is to create trees from a starting root node and branch out to test on the most informative, or most class-discriminative features first. C4.5 uses information gain (IG) (or a variant, gain ratio) to estimate the most informative features. IG is an estimate of how much information a feature contributes to predicting the correct class label. 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 (equation 1):

$$IG_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v) \qquad [1]$$

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. The probabilities are estimated from relative frequencies in the training set.

IGTREE also uses IG. The key difference between IGTREE and C4.5 is that 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. This means that throughout any particular level of the trie, the same feature is tested. This is not the case with C4.5, which recomputes

4. An implementation of IGTree is freely available as part of the TiMBL software package, which can be downloaded from http://ilk.uvt.nl/

IG at every node to determine the next most important feature. This difference makes IGTREE computationally less complex than C4.5 in tree induction.

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 and Van den Bosch, 1992; Daelemans, Van den Bosch, and Zavrel, 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 (for which we will provide empirical back-up evidence later), 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 $O(n\ f)$ of IB1-IG, given that in the typical case n is much higher than f or v (Van den Bosch, 1997).

The example in Figure 1 illustrates the construction of a trie for the case of word prediction limited to predicting *say* or *says* in the seven sentences displayed on the left. These seven sentences are compressed into a trie built up of nodes (the oval units) and branches (with a rectangular label on them). The root (top) node represents all seven examples, hence is labeled with *say* (which occurs four times, versus three times for *says*). The branches connecting the root node to the second-level nodes represent all the pronouns in the word to the left, each denoting unambiguously whether the outcome is *say* or *says*. Third-level nodes matching on the one possible right word (*so*) are superfluous; so are the second-level nodes that predict the same class as the root node. The resulting trie is therefore only the gray part, consisting of four nodes and three branches. Essentially, this trie encodes that the outcome is *say*, unless the word to the left is *he*, *she*, or *it*, in which case it is *says*.

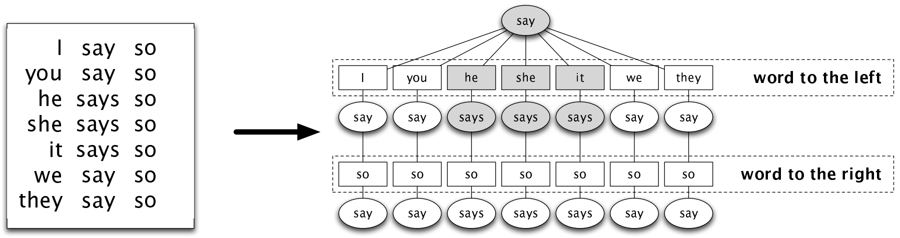

Figure 1. *The disambiguation of* say *and* says *in a limited context of one word to the left (*I, you, he, she, it, we, *or* they*) and one word to the right (*so*), compressed into a trie (right), of which only the gray nodes and branches need to be retained*

2.4. *Evaluation metrics*

In our experiments we carry out the following measurements:

1) Token prediction accuracy — the percentage of correctly predicted tokens in test data.

2) Token prediction speed — the number of test token predictions per second.

3) Number of nodes — the number of trie nodes, including the root node and all other non-ending nodes.

4) Discrete perplexity — a new metric explained in more detail below.

We introduce *discrete perplexity* (DP) as an alternative to the standard notion of perplexity (Roukos, 1996; Jelinek, 1998), which has a strictly probabilistic interpretation. A useful informal definition of perplexity is the number of different tokens that a language model holds as the likely candidate tokens following a token sequence, or in the middle of a sequence. A naive unigram language model, that knows about the frequencies of tokens, but not of their co-occurrence, would expect any word at any position proportional to their likelihood. This "level of surprise" is usually estimated as two to the power of the word-level entropy of a corpus. For example, the word-level entropy of the TEST-REUTERS test set (R) in which W is the set of all tokens occurring in R, $H(R) = -\sum_{w \in W} P(w) \log_2 P(w)$, is 10.37, which makes its perplexity $2^{10.37} \approx 1319$. Likewise, the entropy of the TEST-BROWN test set is 9.94, thus its perplexity 982. Effective stochastic language models (e.g. trigram models with back-off smoothing) are known to bring the perplexity levels on the full Brown corpus down to figures such as 242 (Roukos, 1996).

The problem with adopting the probabilistic perplexity metric in our experiments is that the discrete output of IGTREE does not have a probabilistic interpretation. If IGTREE predicts a word, it does so either because it has found a fully matching variable-width path in its tree leading to a leaf node, or because it returned the class label stored at the last non-ending node it visited before it found no match on the continuing branches. In both cases, the returned class label denotes the majority class in the data subset represented at the final node, often based on just a handful or a pair of locally neighboring training examples, and therefore quite remote from a reliable probabilistic interpretation.

Nevertheless we aim to have a measurement of the reduction in "surprise", or perplexity, that an IGTREE-based classifier is able to attain. Our proposal involves the mapping of the output of a classifier on test data by changing, in a supervised post-hoc evaluation similar to computing accuracy, all incorrectly predicted tokens in the output to a single string, e.g. "XXXXX", but leaving all correctly predicted tokens as they are. If the classifier would predict every word incorrectly, it would produce a text consisting only of "XXXXX", which has a word-level entropy of 0.0. This means there has been no decrease whatsoever in the "surprise", or the perplexity, of the classifier as compared to the unigram perplexity. At the other end of the spectrum, if the classifier would predict all tokens correctly, it would produce a text identical to the original one, with an identical entropy; there would be no surprise left, hence the classifier would have reduced the perplexity to zero.

More formally, we define the discrete perplexity, DP, of a text T_P predicted by a word-prediction classifier and processed as described above, where T_T is the target text to be predicted, and $H(T_T)$ and $H(T_P)$ are the word-level entropies of the two texts, as

$$DP(T_P) = 2^{(H(T_T)-H(T_P))} \qquad [2]$$

3. Word 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 2. The best accuracy observed is 42.2% with 30 million training examples, on TEST-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 TEST-BROWN are slightly higher than on TEST-ALICE, but the difference is small; at 30 million training examples, the accuracy on TEST-ALICE is 12.6%, and on TEST-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 TEST-REUTERS increases by a constant rate of about 8%, while the increases on TEST-ALICE and TEST-BROWN are both about 2% at every tenfold.

We subsequently performed experiments with muting, i.e., the exclusion of ex-

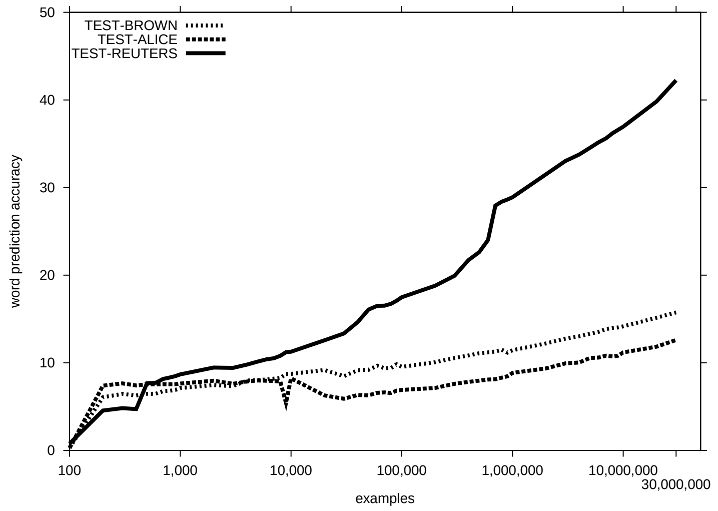

Figure 2. *Learning curves of word prediction accuracies without muting, of* IGTREE *trained on* TRAIN-REUTERS, *and tested on* TEST-REUTERS, TEST-ALICE, *and* TEST-BROWN

amples in the training set of words occurring below a frequency threshold. Figure 3 displays the learning curves on TEST-REUTERS without muting, compared to muting levels 10, 100, and 1,000. The shapes of the learning curves bear a clear similarity; they exhibit the same log-linear increase after about 50,000 examples. The performances at muting levels 10 and 100 are slightly higher than the performances without muting at the same amounts of training examples, while the performance with muting level 1000 only slightly lags behind, and appears to catch up with the other curves with more training examples.

3.2. *Memory requirements, classification speed, and discrete perplexity*

The most noticeable difference between the experiments with and without muting is that the decision trees built in the muting experiments are smaller. Moreover, the numbers of nodes in all different variants also exhibit an interesting linearity with respect to the number of training examples, as suggested earlier by the asymptotic complexity order $O(n)$, where n is the number of training instances. Figure 4 illustrates this relation. The amounts of nodes per muting level related to the number of training examples appear to lie at almost constant factors below 1.0 (i.e., less than one node per training example); for example, at muting level 10 the factor appears to be 0.53 (i.e., at 5 million instances, $5 \times 0.53 \approx 2.65$ million nodes are created)[5].

5. In the used implementation, on a 32-bit processor one node takes 20 bytes to store (two four-byte pointers providing links to other nodes, and three four-byte pointers linking to information

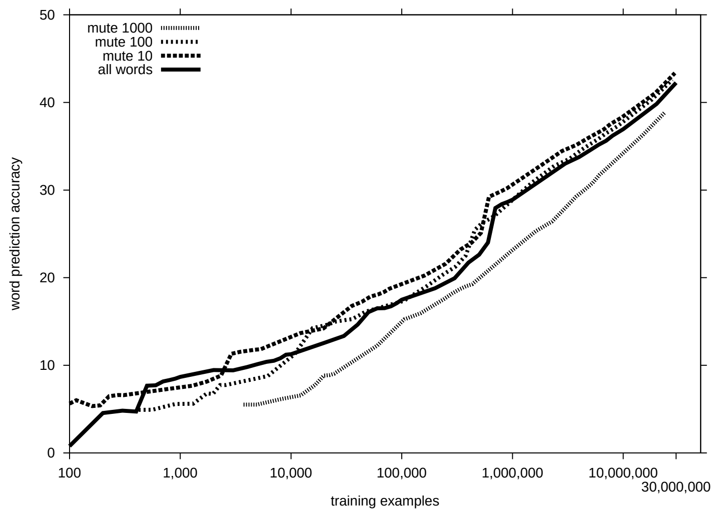

Figure 3. *Learning curves of word prediction accuracies at four muting levels, of* IGTREE *trained and tested on* Reuters *data*

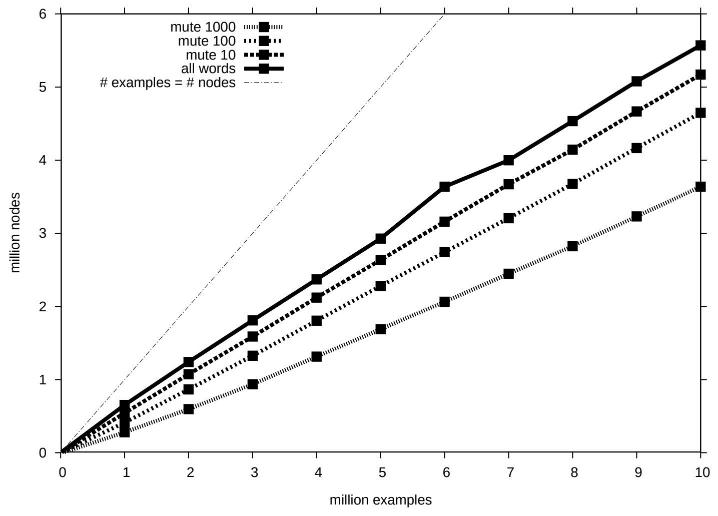

Figure 4. *Numbers of decision-tree nodes at up to 10 million examples, at the four muting levels*

A factor in classification speed is the average branching factor. Conceivably, the word prediction task can lead to a large branching factor, especially in the higher levels of the tree. However, not every word occurs before or after every other word in finite amounts of text. Furthermore, there are relatively strong constraints (of collocational, syntactic, and semantic nature) in a language such as English as to which words can be neighbors. 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 in the no-muting condition 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 and on the task, we observe quite favorable classification speeds. Figure 5 displays the various speeds (in terms of the number of test tokens predicted per second) attained on the three test sets[6]. Obviously, higher speeds correlate with lower prediction accuracies, but the best performances 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 (TEST-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 graphs in Figure 5 do not exhibit a constant log-log decrease; rather, the largest slowdown occurs between 10,000 and 1,000,000 training examples, after which the speed decrease settles on a lower rate. While trees grow linearly, and performance grows log-linearly, the speed of classification slowly diminishes at decreasing rates (note the logarithmic scale of the y-axis of Figure 5).

To illustrate our final measurement, Figure 6 displays the learning curves of the experiment without muting, testing on all three test sets, in terms of discrete perplexity. On TEST-REUTERS, made up of the same type of newswire data as the training set, a strong decrease can be observed from the original level of 1319 down to 57, a level of probabilistic perplexity typically attained by probabilistic language models on domain-specific texts (Roukos, 1996). On TEST-ALICE, which has a low discrete perplexity to begin with, the relative decrease in discrete perplexity is much smaller (57%) than with TEST-REUTERS (96%). On TEST-BROWN the relative decrease after 30 million training tokens is 64%, its discrete perplexity at that point being 351, somewhat higher than, but in the same order of magnitude as the aforementioned trigram model perplexity of 242 on the Brown corpus (Roukos, 1996), which is interesting given that our training set is not composed of generic English texts.

---

on the feature value that labels the branch leading to the node, the class label, and frequency information); consequently, this example tree costs a mere 50 megabytes to store.

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

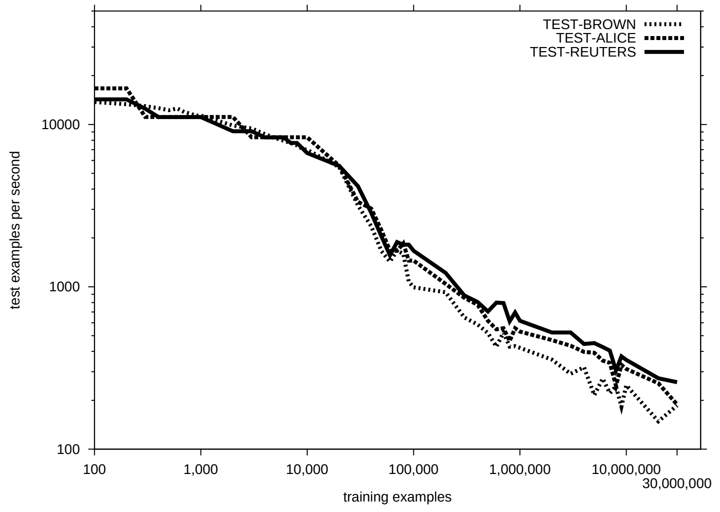

Figure 5. *Word prediction speed, in terms of the number of classified test examples per second, measured on the three test sets, with increasing training examples. Both axes have a logarithmic scale*

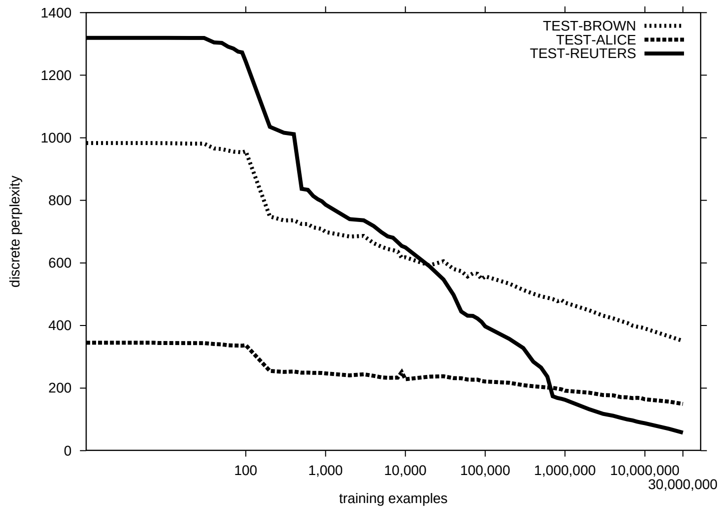

Figure 6. *Learning curve in terms of discrete perplexity, computed on the output of* IGTREE *trained on* TRAIN-REUTERS *and tested on* TEST-REUTERS, *without muting*

# Training examples	Context	TEST-REUTERS	TEST-ALICE	TEST-BROWN
1 million	Left	22.8	7.1	8.9
	Left & right	28.9	8.8	11.4
10 million	Left	28.0	8.0	9.9
	Left & right	36.9	11.2	14.2

Table 3. *Word prediction accuracies on the three test sets at 1 million and 10 million training examples from* TRAIN-REUTERS, *without muting, in the experimental variant with left context only, and with both left and right context*

3.3. *Experiments with left context only*

As stated earlier, another realistic variant of the word prediction task (e.g. for sentence completion or directed language modeling) is to predict a word given only the left context leading up to the word. We performed a series of experiments identical to the experiments described above, except for the fact that no right context was included in the examples presented to the machine learner. A straightforward assessment of the learnability of the "left-context-only" task is that it offers half the information available in the task with both left and right context available, so that one might expect lower prediction accuracies on this task compared to the other task at the same amount of training material. In fact, we observed that the drop in performance is reasonably limited. For illustration, Table 3 displays the word prediction accuracies on the three test sets when training on 1 million (top) and 10 million examples (bottom) from TRAIN-REUTERS in both experimental conditions.

The learning curves of the "left-context-only" experiments in log-log space turn out to exhibit roughly the same log-linear shape and steepness as the curves of the experiments with both contexts available, and are merely horizontally shifted to the right. Apparently, the lack of information caused by the absence of right context is only realized as a relative lag in performance of about four to eight percentage points, and having more training examples compensates for the lack of information: with 10 million training examples, roughly similar word prediction accuracies are attained with left context only as are obtained with left and right context at 1 million examples.

4. Confusibles

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. Learning to predict content words is furthermore hampered by the fact that examples of these content words tend to be sparse – they will

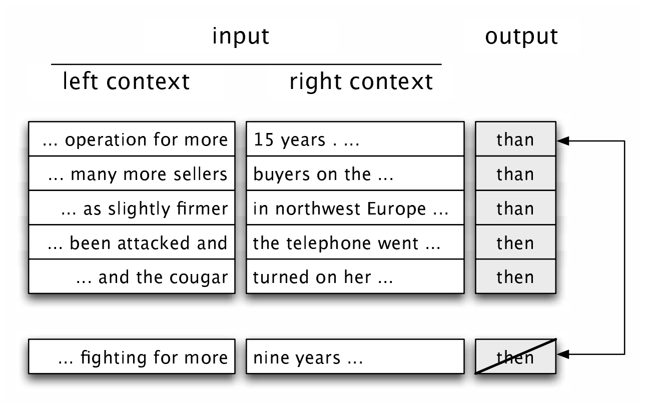

Figure 7. *Illustration of confusible disambiguation for correction purposes. The bottom test example, with the incorrect* then *in focus, finds a best match on the first of the five training examples, suggesting the alternative* than *instead*

typically be in the tail of the Zipfian distribution. In contrast, certain function words tend to be predictable due to the positions they take in lexico-syntactic phrases and attachment structures; 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, confusible errors between frequent forms occur relatively frequently. Examples of these so-called confusibles in English are *there* versus *their* and the contraction *they're*; or the duo *than* and *then*. Confusibles can arise from having the same pronunciation (homophones), or having very similar pronunciation (*country* or *county*) or spelling (*dessert*, *desert*), having 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, or from accidental mistypings, e.g. caused by keyboard proximities.

Distinguishing between confusibles 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 confusibles. Figure 7 illustrates this checking-and-correction procedure. Given a new sentence, with the fragment . . . *fighting for more then nine years* . . ., a matching pattern is found in the (decision-tree compression of the) training examples that is labeled with *than*, suggesting a correction of *then* by *than*.

We performed a series of experiments on disambiguating nine frequent confusibles in English; these are also investigated in (Golding and Roth, 1999). We adopted an experimental setting in which we use the same experimental data as before (i.e., training on TRAIN-REUTERS, and testing on TEST-REUTERS, TEST-ALICE, and TEST-BROWN), in which only examples of the confusible words are drawn – note that we ignore possible confusible 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. This large corpus of 1,096,950,281 tokens offers about ten times as many examples of the confusible words as TRAIN-REUTERS.

As a first illustration of the experimental outcomes, we focus on the three-way confusible *there – their – they're* for which we trained one classifier, which we henceforth refer to as a confusible expert. The learning curve results of this confusible expert are displayed in Figure 8 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 generic word prediction experiments, and to the observation by (Banko and Brill, 2001), the learning curves of this confusible 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.

In the bottom of the same Figure 8 we have also plotted the word prediction accuracies on the three words *there*, *their*, and *they're* attained by the generic word predictor (without muting) described in the previous section, by analyzing the output of this predictor on the three test sets. The accuracies, or rather recall figures (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 confusible disambiguation task. Clearly, *there*, *their*, and *they're* are hard to predict for the generic word predictor (even though it does improve with more training material), while it is quite capable of distinguishing between them in isolation.

Table 4 presents the experimental results obtained on nine confusible sets when training and testing on Reuters material. The nine sets are part of the sets studied in (Golding and Roth, 1999). The third column lists the accuracy scores of the generic word prediction system (without muting) at the maximal training set size of 30 million labeled examples; this is the percentage of cases that the generic prediction system has to predict one of the words in the confusible pair or triple, and actually predicts it correctly. The third and the fourth columns list the accuracies attained by the confusible expert for the particular confusible pair or triple, measured at 30 million training examples, from which each particular confusible expert's examples are extracted. The amount of examples varies for the selected confusible sets, as can be seen in the second column.

Scores attained by the generic word predictor on these words vary from below 10%

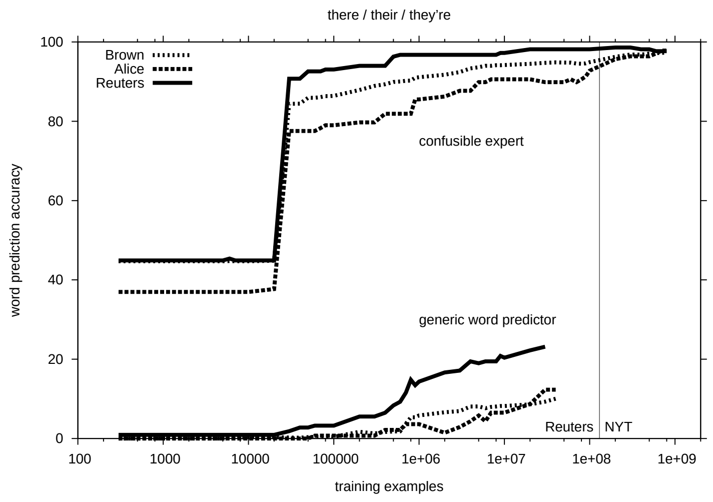

Figure 8. *Learning curves in terms of word prediction accuracy on deciding between the confusible pair* there, their, *and* they're, *by* IGTREE *trained on* Reuters *data, and tested on* TEST-REUTERS, TEST-ALICE, *and* TEST-BROWN, *without muting. The top graphs are accuracies attained by the confusible expert trained on confusible examples only; the bottom graphs are attained by the generic word predictor trained on* TRAIN-REUTERS *until 130 million examples, and on* TRAIN-NYT *beyond (marked by the vertical bar)*

		Accuracy (%) by	
Confusible set	Number of examples	generic word prediction	confusible 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 4. *Disambiguation scores on nine confusible set, attained by the generic word prediction classifier trained on 30 million examples of* TRAIN-REUTERS, *and by confusible experts on the same training set. The second column displays the number of examples of each confusible set in the 30-million word training set; the list is ordered on this column*

	Accuracy on test set (%)		
Confusible set	TEST-REUTERS	TEST-ALICE	TEST-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 5. *Disambiguation scores on nine confusible set, attained by confusible experts trained on examples extracted from 1 billion words of text from* TRAIN-REUTERS *plus* TRAIN-NYT*, on the three test sets*

for relatively low-frequent words to around 60% for the more frequent confusibles; the latter numbers are higher than the overall accuracy of this system on TEST-REUTERS. Nevertheless they are considerably lower than the scores attained by the confusible disambiguation classifiers. While the generic word predictor is trained on all 30 million examples, each confusible classifier is trained only on the subset of examples labeled with one of their particular confusible words. Most of the confusible 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 confusible experts. Table 5 displays the nine confusible 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 TEST-ALICE and TEST-BROWN do not seriously lag behind those on TEST-REUTERS; some are even better – for instance, the four least frequent confusibles are predicted perfectly correctly in TEST-ALICE.

5. Related work

As remarked in the cases reported in the literature directly related to the current article, discrete, classification-based word prediction is a core task to natural language processing, and one of the few that takes no morpho-syntactic or semantic 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). It is almost puzzling why it is not a focus in current natural language processing research, as it has been, albeit in a probabilistic interpretation, in language modeling for automatic speech recognition (Jelinek, 1998) - it may have been out of focus due to the prohibitively large feature and class spaces; exactly because of this reason it is an ideal task to consider as a case in scaling studies.

Given that scaling issues are becoming increasingly important in natural language processing research, our view is that word prediction may be to natural language processing what language modeling is to speech processing: a core task that, when performed well, paves the way for dealing with low-frequency words and errors in text in further processing steps such as syntactic and semantic parsing, potentially surpassing the need for intermediate abstraction levels directly above the word level such as part-of-speech tagging (Van den Bosch and Buchholz, 2002). The latter goal of providing alternative, implicit ways of learning about the structure of language from the bottom up, is an echo of classical ideas of Zellig Harris (Harris, 1957; Harris, 1968), via which the work reported here inherits a link to current work in grammar induction (Adriaans *et al.*, 2004).

More directly, the work of Even-Zohar and Roth remains the closest nearest neighbor of this article (Even-Zohar and Roth, 2000). Two of their arguments deviate from ours. First, inspired by analogous work in probabilistic language modeling (Chelba and Jelinek, 1998), they argue that syntactic features are necessary for generic classification-based word prediction. Although we have not made the comparison directly (we may do this in future work), we have set out to show that word prediction can be learned on the basis of examples that represent merely contexts of word and punctuation tokens - arguably, given enough examples, words can take over the role of part-of-speech tags implicitly, as we argued and showed earlier in (Van den Bosch and Buchholz, 2002). Second, Even-Zohar and Roth argue that it is *necessary* to focus the attention of the classifier to limited sets of confusibles, which we see as an *optional* specialization that for practical reasons (given the current capacity of our computers) still performs better than generic word prediction. They experiment on pairs of verbs from the Wall Street Journal Penn Treebank corpus that are about equally likely based on their frequency of occurrence and part-of-speech (e.g. *make*/*sell*), which deviates from our more practically-driven definition of confusible as also used by (Golding and Roth, 1999; Banko and Brill, 2001). Even-Zohar and Roth use small amounts of training and test data (together less than 100,000 examples), and report word error rates of close to 10% (90% word accuracy), counter to about 65% word error rate (35% word accuracy) when training and testing on all verbs simultaneously (Even-Zohar and Roth, 2000).

The papers by Golding and Roth, and Banko and Brill on confusible 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 confusible correction task to illustrate scaling issues, as we have. Golding and Roth illustrate that multiplicative weight-updating algorithms such as Winnow (Littlestone, 1988) 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. Banko and Brill, on their part, use the confusible task as a "fruit fly" task to illustrate the log-linear growth effect in accuracy, when the number of training examples is exponentially grown. Their point is to show that a lot may change in performances when instead of the seemingly high number of a million examples (which still is the upper bound for a lot of work in machine learning applied to natural language processing) a multiple amount of examples becomes available (Banko and Brill, 2001). The paper provides no exact information on the confusibles that are actually tested; they also present learning curve graphs with a logarithmic x-axis up to one billion instances, assumedly to indicate that they sampled from a corpus of up to one billion tokens to create smaller confusible example subsets.

More generally, as a precursor to the above-mentioned work, confusible 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). As a side note, confusible correction is often referred to as context-sensitive spelling correction (Golding and Roth, 1999), while it is obvious that non-word spelling correction (i.e. correction of typos that lead to non-existing words) can greatly benefit from context as well (Kukich, 1992; Reynaert, 2004; Reynaert, 2005).

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 confusible disambiguation, a specialization of word prediction focusing on small sets of confusible 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 confusible disambiguation accuracies of well above 90%.

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.

In addition, we observed that muting (discarding low-frequency words from the training data) does not improve word prediction results, but at level 10 (ignoring words below that frequency of occurrence) it yields the same accuracies with smaller decision trees. This is an indication that the performances of the word prediction systems are largely based on their ability to predict high-frequent words, i.e. function words, correctly at certain points.

Also, we observed that predicting words solely on the basis of left context does lead to lower word prediction accuracies as compared to the variant in which left and right contexts are available. However, the learning curves in the "left-context-only" condition are as log-linear and steep as the learning curves observed in the experiments with left and right context available; the lack of right context can be compensated by training on more "left-context-only" examples.

Storing the trees is favorably linear in the number of training examples, at a rate lower than $1:1$, and close to one node for every two training examples in the worst case (without muting). We also measured classification speeds, and noted that with more training examples and linearly growing decision trees, classification speeds decrease, with an increasingly slower rate of decrease; the slowest measured speed of a trained word prediction classifier is still over one hundred words per second.

Accuracies on test data different from the training material, the full text of *Alice's Adventures in Wonderland* novel and the Brown corpus part of the Penn Treebank, are markedly lower than those on test data from Reuters. Scores on different texts peak at 15.8% word prediction accuracy, indicating that the high scores on TEST-REUTERS are due to the large amount of overlap in phrases in *Reuters* newswire articles over time. A likely explanation for the large difference is that the overlap in content words in the training and test data from the *Reuters* corpus is much higher than the overlap between TRAIN-REUTERS on the one hand, and TEST-ALICE and TEST-BROWN on the other hand. The 12-16% word prediction accuracies on the latter test data are likely to stem from the correct identification of certain more frequent tokens, i.e. function words and punctuation markers, while the gap between these scores and the 43% on TEST-REUTERS is likely to be due to correctly predicted relatively lower-frequency tokens, i.e. newswire-specific content words.

We have studied a deliberately simple method, IGTREE, which performs lossless trie compression (Knuth, 1973), and which operates on examples which represent mere sequences of words. A severe weakness of IGTREE is its inability to match an unseen example at several lower-importance features when a feature with a larger estimated importance mismatches. Its decision-tree strategy dictates that it matches features in a strict order of importance, returning a best guess as soon as it cannot find a match on its next feature-value test. The sooner it has to make a guess, the weaker the guess (Daelemans, Van den Bosch, and Zavrel, 1999). Future work may focus on softening the search strategies more in the direction of k-nearest neighbor classification (Daelemans, Van den Bosch, and Zavrel, 1997b), however this will lead to speed loss. In future work we intend to investigate a parallelization of the tree construction process on multi-processor hardware, e.g. by parallelizing tree induction through treating every second-level node as a root node.

The implications of this study are twofold. On the one hand, a word prediction system based on the method proposed here could be a highly efficient and valuable module in authoring environments in which new documents are like older documents. The module could play a role in suggesting words or completing phrases, or in spelling or grammar checking. On the other hand, the quality of prediction breaks down easily when applied to diverging types of text. The present study offers only slim indications that the proposed method would perform reasonably well when applied to diverging types of text when trained on (a lot) more training material.

However, as an alternative to generic word prediction, evidence is shown (echoing demonstrations in earlier work) that particular confusibles of relatively high frequency can in fact be disambiguated by "confusible experts", focused classifiers trained exclusively on examples of confusible pairs or triples, which could be used for correction purposes in text processing environments (and thus have a more limited applicability than generic word prediction systems). The disambiguation accuracies on test data of these experts are mostly above 90%, regardless of the type of test set, and are already attained at relatively low amounts (e.g. thousands) of training examples.

To generalize the confusible task, we intend to focus some future work on the development and software engineering of an automatically-generated ensemble of confusible experts, in the line of (Huang and Powers, 2001), that would form a proofing tool plugin for a word processor, where the sets of confusibles are automatically detected in a language-independent fashion, and where the full ensemble of confusible experts represents an optimal trade-off in memory usage, speed, and word prediction accuracy.

Acknowledgements

The author would like to thank Martin Reynaert, François Yvon, and three anonymous reviewers for their valuable comments and remarks on earlier versions. This work was funded by NWO, the Netherlands Organisation for Scientific Research.

7. References

Adriaans P., Fernau H., de la Higuera C., van Zaanen M., " Introduction to the special issue on grammar induction", *Grammars*, vol. 7, p. 41-43, 2004.

Banko M., Brill E., " Scaling to Very Very Large Corpora for Natural Language Disambiguation", *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, p. 26-33, 2001.

Breiman L., Friedman J., Ohlsen R., Stone C., *Classification and regression trees*, Wadsworth International Group, Belmont, CA, 1984.

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

Chelba C., Jelinek F., " Exploiting Syntactic Structure for Language Modeling", *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Montréal, Quebec, Canada*, p. 225-231, 1998.

Daelemans W., Van den Bosch A., " Generalisation Performance of Backpropagation Learning on a Syllabification Task”, *in* M. F. J. Drossaers, A. Nijholt (eds), *Proceedings of TWLT3: Connectionism and Natural Language Processing*, Twente University, Enschede, p. 27-37, 1992.

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

Daelemans W., Van den Bosch A., Zavrel J., “ A feature-relevance heuristic for indexing and compressing large case bases”, *in* M. Van Someren, G. Widmer (eds), *Poster Papers of the Ninth European Conference on Machine Learing*, University of Economics, Prague, Czech Republic, p. 29-38, 1997b.

Daelemans W., Van den Bosch A., Zavrel J., “ Forgetting exceptions is harmful in language learning”, *Machine Learning, Special issue on Natural Language Learning*, vol. 34, p. 11-41, 1999.

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

Garay-Vitoria N., González-Abascal J., “ Intelligent word-prediction to enhance text input rate”, *Proceedings of the 2nd International Conference on Intelligent User Interfaces*, p. 241-244, 1997.

Golding A. R., “ A Bayesian hybrid method for context-sensitive spelling correction”, *Proceedings of the ACL-95 3rd Workshop on Very Large Corpora*, p. 39-53, 1995.

Golding A., Roth D., “ A Winnow-Based Approach to Context-Sensitive Spelling Correction”, *Machine Learning*, vol. 34, n° 1–3, p. 107-130, 1999.

Harris Z. S., “ Co-Occurrence and Transformation in Linguistic Structure.”, *Language*, vol. 33, n° 3, p. 283-340, 1957.

Harris Z. S., *Mathematical structures of language*, Wiley, 1968.

Hiemstra D., Using language models for information retrieval, PhD thesis, University of Twente, 2001.

Huang J. H., Powers D. W., “ Large scale experiments on correction of confused words”, *Australasian Computer Science Conference Proceedings*, Bond University, Queensland AU, p. 77-82, 2001.

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

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

Kukich K., “ Techniques for Automatically Correcting Words in Text”, *ACM Computing Surveys*, vol. 24, n° 4, p. 377-439, 1992.

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

Littlestone N., “ Learning Quickly when irrelevant attributes abound: A new linear-threshold algorithm”, *Machine Learning*, vol. 2, p. 285-318, 1988.

Mangu L., Brill E., “ Automatic rule acquisition for spelling correction”, *Proceedings of the International Conference on Machine Learning*, p. 187-194, 1997.

Marcus M., Santorini S., Marcinkiewicz M., “ Building a Large Annotated Corpus of English: the Penn Treebank”, *Computational Linguistics*, vol. 19, n° 2, p. 313-330, 1993.

Martin W., Al B., van Sterkenburg P., “ On the processing of a text corpus: From textual data to lexicographical information”, *in* R. Hartman (ed.), *Lexicography: Principles and Practice*, Applied Language Studies Series, Academic Press, London, 1983.

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

Reynaert M., “ Text-induced spelling correction”, *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switserland, p. 117-124, 2004.

Reynaert M., Text-induced spelling correction, PhD thesis, Tilburg University, 2005.

Roukos S., “ Language representation”, *in* R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue (eds), *Survey of the state of the art in human language technology*, Chapter 1.6, Center for Spoken Language Understanding, 1996.

Van den Bosch A., Learning to pronounce written words: A study in inductive language learning, PhD thesis, Universiteit Maastricht, 1997.

Van den Bosch A., Buchholz S., “ Shallow parsing on the basis of words only: A case study”, *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, p. 433-440, 2002.

Wood M., Syntactic pre-processing in single-word prediction for disabled people, PhD thesis, University of Bristol, 1996.

Wu D., Sui Z., Zhao J., “ An information-based method for selecting feature types for word prediction”, *Proceedings of the Sixth European Conference on Speech Communication and Technology, EUROSPEECH’99*, Budapest, p. 472-479, 1999.

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

Zipf G. K., *The psycho-biology of language: an introduction to dynamic philology*, The MIT Press, Cambridge, MA, 1935. Second paperback edition, 1968.