Automatic Rule Acquisition for Spelling Correction

Lidia Mangu and Eric Brill
Computer Science Dept.
Johns Hopkins University
Baltimore, MD 21218
{lidia, brill}@cs.jhu.edu

Abstract

This paper describes a new approach to automatically learning linguistic knowledge for spelling correction. A major feature of this approach is the fact that the acquired knowledge is captured in a small set of easily understood rules, as opposed to a large set of opaque features and weights. A perspicuous representation is advantageous in order to best exploit human intuition to understand and improve upon the acquired knowledge of the system.

1 Introduction

With the recent availability of large on-line text corpora, it has become possible to explore techniques for automatically learning linguistic information directly from text, instead of laboriously hand-crafting rules to encode this information. When a legitimate word is replaced by a non-word (such as **teh** for **the**), absence of a string from a dictionary is often sufficient for detecting a mistake and the edit distance between the non-word and legal words in a dictionary is often sufficient for correcting the mistake. However, a large class of errors involve misspellings that result in valid words. This can be the result of a typographical error (e.g. **dog** vs. **dot**), or from a person being unsure or incorrect about a particular word usage (e.g. **among** vs. **between**). Studies have shown that errors of this type account for anywhere from 25% to 50% of word-based errors in a text, depending on the application (Kukich, 1991). Such errors seem to require both syntactic and semantic information from the surrounding context in order to perform accurate correction.

For example, in the sentence:

I am just a human begin/being.

we know the correct word is **being** because the preceding word is **human**. In the sentence:

My paycheck is bigger than/then yours.

we know the correct word is **than** because the word **then** rarely follows an adverb. In the sentence:

The principal/principle left the school after dark.

we know the correct word is **principal** because principals are associated with schools.

One approach to solving this problem would be to rely on our linguistic intuitions to manually generate a set of rules such as those above. However, this can be time-consuming. Since the rules of language are vast and idiosyncratic, a person would likely miss important and powerful rules if relying solely on intuition. Portability is also an issue, as both the rules and the likely confusable words would change when moving from one domain to another. For instance, the rules that one would write to predict the word **mass** would be very different for a physics text than for a liturgical document. It would be desirable to automatically train a system when moving to a new domain rather than having to manually regenerate a set of applicable rules.

A number of approaches have been recently proposed for automatically training a system to perform this task. These methods have the advantage of being trainable as opposed to being manually derived. While they achieve high rates of accuracy, they sacrifice the clarity of a set of rules such as those given above. Instead of learning intuitive rules, they extract large sets of opaque features and weights. This has the disadvan-

tage of making it hard to use one's linguistic knowledge to understand what was learned, in order to gain insight that would help improve the learning process. In addition, a small and easily understood set of acquired knowledge is desirable as it allows for a better integration of both manual and automatic approaches.

In this paper we present a method for combining the clarity of manually constructed rule-based systems with the trainability of the automated systems. After describing the specific problem we are addressing and past approaches at applying machine learning techniques to this problem, we describe a method that is able to automatically learn simple, small rule sets, while achieving accuracy comparable to the very best of the past machine learning approaches.

2 Previous work

An initial approach to automatic acquisition for context-based spelling correction was a statistical language-modeling approach using word and part of speech *n-grams* (Atwell and Elliott, 1987; Gale and Church, 1990; Mays et al., 1991; Church and Gale, 1991) in which a low-probability word sequence is used to detect real-word errors and high-probability ones are used to rank correction candidates. For example, in the sentence *"The cat mowed loudly"* the probability $P(mowed|The\ cat)$ is very low, indicating that this is possibly an error. Of words within a small edit distance of **mowed**, $P(meowed|The\ cat)$ would be greatest, indicating that **meowed** was likely the intended word in this sentence. One problem with these models is that they require a large body of text for training the *n-gram* model and huge *n-gram* tables available at run time. In addition, they do not capture long-range dependencies.

A different approach is to treat context-based spelling correction as a problem of ambiguity resolution. The ambiguity among words is modeled by *confusion sets*. Rather than attempting to detect and correct all errors, this approach attempts to choose between commonly confused words (such as **than** vs. **then**). The first efforts within this framework for spelling correction include Bayesian classifiers and decision lists (Gale, Church, and Yarowsky, 1995; Yarowsky, 1994; Golding, 1995). In (Golding , 1995) decision lists are first used to choose the proper word from a confusion set. It was determined that better results can be obtained when evidence is combined in a more powerful way, by taking into consideration not just the strongest piece of evidence but all the available evidence. (Golding, 1995) ran the same experiments with Bayesian classifiers, and achieved a small improvement over decision lists. In (Golding and Schabes, 1996) an extension is made to the Bayesian classifier, where a part of speech tagger is first used to tag the text. When all words in a confusion set differ in part of speech, the tagger is used to determine the most likely tag, and thereby the most likely word, in a context. When words do not differ in part of speech, the original Bayesian classifier is used. This resulted in an improvement over the original Bayesian classifier.

One of the most successful methods for this problem is a *Winnow-based* method (Golding and Roth, 1996), a multiplicative weight-updating algorithm which achieves a good accuracy by being able to handle a large number of features. This method also learns a large set of features with corresponding weights. The Winnow approach for weight training appears to be more effective than the Bayesian classifier, as the Winnow system achieves significantly better performance than the Bayesian classifier + tagger (*TriBayes*), when run on the same training and test suite, and using the same set of features. This is likely due to a combination of factors, including the method for deriving the weights in Winnow, as well as the method used for combining a number of different Winnow-based classifiers through weighted voting.

While these approaches result in systems with high accuracy, their acquired knowledge is represented opaquely in large sets of features and weights. The success of these systems is evidence for the sufficiency of the acquired knowledge in the form in which it is acquired and used. In this paper we explore whether such a representation is also necessary, or whether a method where a small number of simple disambiguation cues are learned can achieve comparable performance.

3 Transformation-Based Learning

Our goal was to determine whether a learning algorithm could be used for spelling correction which could achieve high accuracy while capturing its learned knowledge in a form that could be easily understood. Ideally, we hoped to learn intuitive rules such as those shown in Section 1. One recently proposed approach for rule-learning is transformation-based learning. Transformation-based learning has been applied to a number of natural language problems, including part of speech tagging, prepositional phrase attachment disambiguation, speech generation and syntac-

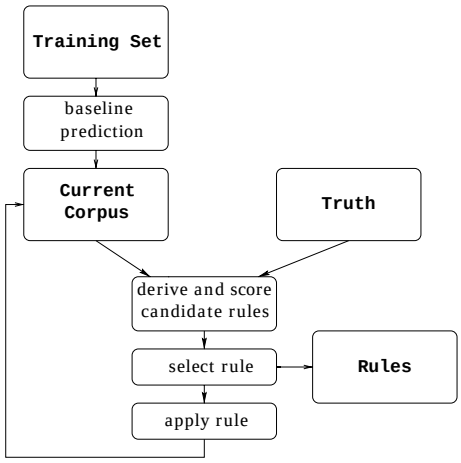

Figure 1: Transformation-based Learning

tic parsing (Brill, 1993; Brill, 1995) often achieving state-of-the-art accuracy while capturing the learned knowledge in a small and easily understood set of rules. Figure 1 shows the learning process.

To fully specify a transformational system, one must specify:

1. A baseline predictor.
2. A set of allowable transformation types.
3. An objective function for learning.

In learning, the training set is first passed through a program which makes an initial baseline prediction. The quality of this baseline corpus can then be measured by comparing it to the truth, according to the specified objective function. The learner then iteratively learns an ordered sequence of rules. A single iteration of the learner consists of doing the equivalent of:

1. For each possible transformation that can be applied:
   (a) Apply that transformation to a copy of the corpus.
   (b) Score the resulting corpus according to the objective function.
2. Pick the transformation that achieves the best score.
3. Append that transformation to the end of the transformation list.
4. Apply it to the training corpus
5. Go To 1

Learning ends when no transformations can be found whose application results in an improvement to the training corpus. In actuality, there may be an extremely large set of potential transformations, and applying and scoring each would be infeasible. Instead, we can do the equivalent to this in a very efficient way by using a data-driven algorithm (Brill, 1995).

The result of learning is an ordered list of transformations. To apply the learned transformations to new text, that text is first processed by the baseline predictor. Then each rule is applied, in order, everywhere it can be applied. It is important to stress that the rule sequence is applied strictly deterministically: the first rule on the list is first applied and then thrown away, then the second rule is applied and thrown away, and so on until the last rule in the list has been applied.

A few points are worth mentioning about transformational systems. First, such rule sets are actually processors and not classifiers. Given a corpus in any state of annotation, a rule processes the corpus to change the state of annotations. This means that any baseline predictor can be used, including for example another sophisticated spelling correction program, in which case the learner will learn transformations to correct errors made by this system. Second, for a fixed set of features, transformation lists are more powerful than decision trees in what they can learn (Brill, 1995). Finally, unlike both decision trees and decision lists, a rule in the list is able to correct mistakes made by applying previous rules.

3.1 Transformation-Based Learning for Spelling Correction

We next describe how to build a transformation-based system for spelling correction. We first have to specify the space of spelling errors we are looking for. We take a number of words which can be confused with each other and cluster them. The result is a collection of *confusion sets.* Given a confusion set C and words w_1, w_2, ...,$w_n \in$ C, the problem then consists of deciding each time a word from C appears in text, which of the words in the confusion set was the intended word. The **baseline predictor** initially assumes the most frequent word in C is the intended word.[1] Thus, we

[1] Interestingly, the predictor that assumes the least frequent word in C is the intended word achieved only slightly worse accuracy in the experiments we performed.

started by replacing each instantiation of a word in C with the most frequent word in C.

The **allowable transformations** in these experiments are described by the following templates:

- Change word w_1 to w_2 if:
  (a) the word W occurs within $\pm$ k words of w_1 (*co-occurrences*)
  (b) a specific pattern of up to l contiguous words and/or part-of-speech tags occurs around w_1 (*collocations*)
  (c) a specific pattern of noncontiguous words and/or part-of-speech tags occurs around w_1(*collocations with wildcards*)

An example of a rule of type (a) is:[2]

Change the word from **principle** to **principal** if the word **school** appears within the proximity window.

An example of a rule of type (b) is:

Change the word from **piece** to **peace** if the word **world** immediately precedes it.

An example of a rule of type (c) is:

Change the word from **except** to **accept** if the word three before is **he** and the immediately preceding word is **not**.[3]

The **objective function** for evaluating transformations is simply classification accuracy. At each site where the current prediction is not correct, the rule templates are used to form rule candidates for correction. This process identifies all the rules generated by the template set that would have a positive effect on the current assignment. Those candidate rules are tested against the rest of the training set, to identify at how many locations they caused *negative* changes. Each rule is assigned a *score* based on the number of *positive* and *negative* changes caused by applying the rule. The rule with the highest net score is written out as the first rule in the learned sequence and is applied to the corpus. The same learning process is then repeated on the transformed corpus. The entire process leads to an ordered sequence of rules.

[2]Note that the learner is only provided with transformation **types**, and so also could potentially learn nonsensical rules such as: Change the word from **peace** to **piece** if the word **it** appears within the proximity window.

[3]e.g. He {does/did/can/should} not accept.

4 Experiments

For their experiments (Golding, 1995; Golding and Schabes, 1996; Golding and Roth, 1996) used the 1 million word Brown corpus (Kučera and Francis, 1967) and collections of confusion sets which were selected from the list of "Words Commonly Confused" in the back of Random House (Flexner, 1983) on the basis of occurring frequently in the Brown corpus and representing a variety of types of errors. We ran the experiments using the same corpus, and a subset of their collection of confusion sets which is present in all the three papers mentioned above. Even more, we were kindly given the exact same partition of the data into training and test set (80% training/20% test). All these made possible a fair direct comparison with the previous methods. The results for Bayes, TriBayes and Winnow methods which occur in the tables below are taken from (Golding and Schabes, 1996) and (Golding and Roth, 1996). It is worth mentioning that in our experiments we changed the set of features and the annotation we use, but we did not change anything across experiments for the other three methods.

4.1 Use the Same Feature Set and Annotations

In the first experiment we used a set of features containing co-occurrences within a window of 10 words on either side and collocations within two words of the word being checked. This is the same set of features used in both the Bayesian spelling corrector (Golding and Schabes, 1996) and Winnow-based method (Golding and Roth, 1996). In addition, the annotation used for the training and the test set was the same, namely, each word is tagged with the set of all possible part of speech tags. The results are shown in the first column of Table 1. The following tables give a better illustration of the differences between the results of the different methods in the conditions described above. The first table shows a comparison of the average accuracy (AA) and weighted average accuracy (WAA) of the different methods on the confusion sets. The second table shows how many times the rule-based approach (RuleS) was better, worse or equal to the other methods.

	Bayes	TriBayes	Winnow	RuleS
AA	86.94	88.38	93.69	88.73
WAA	87.92	91.15	94.01	88.09

	RuleS >	RuleS <	RuleS =
Bayes	7	5	2
TriBayes	4	6	4
Winnow	1	13	0

4.2 Using a Part of Speech Tagger

In (Golding and Schabes, 1996), it was shown that using a part of speech tagger to provide the proper tags for words in the sentence resulted in an improvement in performance over using a Bayesian classifier without disambiguating the tags. Since taggers are readily available for many domains and languages, and tagging is a very fast process, the additional cost of pre-tagging text with a tagger is minimal. Because of this, we investigated whether using the output of a tagger instead of the set of allowable tags for each word in the sentence being processed could improve our results.

The training set was run through a publicly available part-of-speech tagger (Brill, 1993). In order to tag the test set we have to consider the fact that the identity of the target word has to be established and we can not tag uniquely at spelling-correction time; thus, we substitute each word in the confusion set in the target sentence, then we run the tagger on all the sentences which are obtained and then for each word in the sentence we take the union of the tags. So, in the test set, some words are annotated with sets of tags.

The accuracy for this experiment is shown in the second column of Table 1. We see that the tagger increases the performance of our method.

	Bayes	TriBayes	Winnow	RuleS
AA	86.94	88.38	93.69	91.75
WAA	87.92	91.15	94.01	92.26

	RuleS >	RuleS <	RuleS =
Bayes	12	1	1
TriBayes	10	3	1
Winnow	2	8	4

4.3 Using a Smaller Window and No Tagger

Due to the fact that our method relies heavily on counts, spurious feature matches in wide windows can be very harmful. Therefore, it is likely that the rule-based learner would perform better with a smaller window for context words, as such a window would contain less superfluous information. However, we anticipated that a *larger* window for collocations can improve the performance of our system. Thus for this experiment we considered a window of three words on either side of the target word, and allowed transformations to learn any pattern of words and part of speech tags within that window. This results in an improvement in accuracy over the original, as seen in the following two tables.

	Bayes	TriBayes	Winnow	RuleS
AA	86.94	88.38	93.69	89.85
WAA	87.92	91.15	94.01	90.35

	RuleS >	RuleS <	RuleS =
Bayes	10	2	2
TriBayes	8	5	1
Winnow	1	12	1

4.4 Using a Tagger and a Smaller Window

In the last experiment we used both the tagger and the smaller window. We obtained results which are comparable with the results obtained by the Winnow-based method and better than the results obtained by Bayes or TriBayes. This can be seen in the following two tables and in Table 2.

Even though *collocations with wildcards*[4] were included in our set of features, they were useful in disambiguation only for 2 confusion pairs out of the 14 we considered: {*between, among*} and {*number, amount*}. This is due to the fact that for these two confusion sets there is a considerable overlap in the usage of the words in the set. For example, both *number* and *amount* can be found in the context [the JJ ___ IN][5]. However, if the second word after the preposition is a singular noun then this is a good indication that the word *amount* is the correct one, and if it is a plural noun then it favors the word *number*. The need for *collocations with wildcards* comes from the fact that the word immediately following the preposition does not contain any information for disambiguation.

	Bayes	TriBayes	Winnow	RuleS
AA	86.94	88.38	93.69	92.79
WAA	87.92	91.15	94.01	93.15

	RuleS >	RuleS <	RuleS =
Bayes	13	0	1
TriBayes	10	2	2
Winnow	3	5	6

[4] see Section 3.1

[5] JJ = adjective, IN = preposition

5 Discussion and Conclusions

As shown above, we initially obtained results somewhat worse than those obtained by Winnow on this data set. When we use a smaller window for co-occurrences and a part of speech tagger, we achieve comparable results to those obtained by Winnow without using a tagger. The transformation-based approach loses some accuracy by only using a small number of potential disambiguation cues, and combining these cues in a very simple way. However, the learned information is considerably smaller in size and easier to understand than what is learned with Winnow. In Table 3 we give the number of rules we obtain for each of the confusion sets and compare with the number of parameters used in the Winnow-based method. The transformation-based system learns a very small set of simple rules, compared to a large number of parameters and weights.

Table 3: Comparison of the number of features when using different methods

	Winnow	RuleS
accept, except	849	8
affect, effect	842	8
among, between	2706	85
amount, number	1618	41
being, begin	2219	13
country, county	1213	11
lead, led	833	17
passed, past	1279	18
principal, principle	669	21
quiet, quite	1200	9
raise, rise	621	9
than, then	6813	29
weather, whether	1226	13
peace, piece	992	22

Below we show the entire acquired information for the confusion set {*accept*, *except*}. Initially, we start with the most frequent word: *except* and we learn the following ordered transformational rules :

1. if the previous word is *to* then *except* −>*accept*
2. if the previous word is a modal (MD) then *except* −>*accept*
3. if the previous word is an adverb (RB) then *except* −> *accept*
4. if the previous word is an Wh-pronoun(WP) then *except* −>*accept*
5. if the next two words are proper nouns (NNP) then *accept* −>*except*
6. if the next word is a pronoun(PRP) which is followed by a preposition(IN) then *except* −>*accept*[6]
7. if the next word is an adverb (RB) then *accept* −>*except*
8. if the next word is a conjunction(CC) which is followed by the word *reject* then *except* −>*accept*

As can be seen, the learned information is very easy to understand. For this particular confusion set, this small set of rules obtains the same accuracy as is achieved from Winnow with 849 parameters.

We have presented a method for choosing between confusable words in spelling correction which learns linguistic information in a very compact and readily understood form, with very little sacrifice in performance compared to other systems which learn large sets of parameters. A small and easily understood set of acquired knowledge is desirable as it allows for a better integration of both manual and automatic approaches. For example, one of the rules which is learned is "Change *peace* to *piece* if the previous word is *every*". A human observing this rule could easily reason that the rule "Change... *each*" is also a good rule, but was not learned due to insufficient evidence in the training data. Actually, in a separate experiment we added this rule to the automatically obtained set of rules for the confusion pair {*piece, peace*} and obtained an improvement in accuracy. Another way to combine manual with automatic methods is to stop the process of learning described in Section 2 when the only rules left to be learned have low scores, and to ask people to filter them and decide upon their importance. Low score rules are rules for which the learner does not have enough evidence to know if they are anomalous to the training set or truly useful rules. This is not a hard task for a human and can improve the performance of our system.

In the future, we hope to explore this trade-off between clarity of acquired knowledge and accuracy in this domain further, as well as examining this problem in other domains that can be framed as choosing from a prespecified confusion set.

[6] e.g. "...accept him/PRP in/IN your heart..."

Acknowledgements

Many thanks to Andrew Golding for providing the test and training set he used in the previous experiments and for his very helpful comments. Thanks also to David Yarowsky and John Henderson for valuable discussions and comments concerning this work.

This work was funded by NSF grant IRI-9502312. Both authors are members of the Center for Language and Speech Processing at Johns Hopkins University.

References

Atwell, E. and Elliott, S. (1987). Dealing with ill-formed English text. In *The Computational Analysis of English: A Corpus-Based Approach*. R. Garside, G. Leach, G. Sampson, Ed. Longman, Inc. New York.

Brill, E. (1993). Automatic grammar induction and parsing free text: a transformation-based approach. In *Proceedings of 31st Meeting of the Association of Computational Linguistics*, Columbus, OH.

Brill, E. (1995). Transformation-Based Error-Driven Learning and Natural Language. A Case Study in Part of Speech Tagging. *Computational Linguistics*,**21**(4):543-565.

Church, K. W. and Gale, W. A. (1991). Probability scoring for spelling correction. In *Stat. Comp. 1.*, 93-103.

Flexner S. B., editor (1983). Random House Unabridged Dictionary. Random House, New York. Second edition.

Gale, W. A. and Church K. W. (1990). Estimation Procedures for language context: Poor estimates are worse than none. In *Proceedings of Compstat-90* (Dubrovnik, Yugoslavia), 69-74. New York: Springer-Verlag.

Gale, W., Church, K. and Yarowsky, D. (1995). Discrimination decisions for 100,000 dimensional spaces. *Annals of Operations Research*, 55, 323-344.

Golding, A. (1995). A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*, 39-53. Boston, MA.

Golding, A. and Schabes, Y. (1996). Combining Trigram-based and Feature-based Methods for Context-Sensitive Spelling Correction. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 71-78. Santa Cruz, CA.

Golding, A. and Roth, D. (1996). Applying Winnow to Context-Sensitive Spelling Correction. In *Machine Learning: Proceedings of the 13th International Conference*, 182-190. San Francisco, CA.

Kukich, K. (1991). Automatic spelling correction: Detection, correction and context-dependent techniques. Technical report, Bellcore, Morristown, NJ 07960. Draft.

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

Mays, E., Damerau, F. J. & Mercer, R. L. (1991). Context based spelling correction. *Information Processing and Management*, **27**(5):517-522

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 88-95. Las Cruces, NM.

Table 1: The results obtained when using different sets of features and different annotations. The algorithms were trained on the same 80% of Brown and tested on the other 20%.

Experiment 1: Using the Same Feature Set and Annotations
Experiment 2: Using a Part of Speech Tagger
Experiment 3: Using a Smaller Window
Experiment 4: Using a Tagger and a Smaller Window

Confusion set	Experiment 1	Experiment 2	Experiment 3	Experiment 4
accept, except	88.0	92.0	88.0	92.0
affect, effect	97.9	100	97.9	100
among, between	73.1	80.6	76.4	84.4
amount, number	78.0	86.2	84.5	87.8
begin, being	95.3	97.7	95.3	97.9
country, county	95.2	95.2	96.8	96.8
lead, led	89.8	89.8	87.7	93.9
passed, past	83.7	90.5	87.8	90.5
peace, piece	90	90	88	90
principal, principle	88.2	91.2	88.2	91.2
quiet, quite	92.4	93.9	92.4	93.9
raise, rise	84.6	84.6	84.6	84.6
than, then	92.6	92.8	93.7	96.1
weather, whether	93.4	100	96.7	100

Table 2: The performance of *RuleS* versus *Bayes*, *TriBayes* and *Winnow*. The algorithms were trained on the same 80% of Brown and tested on the other 20%.

Confusion set	Training Cases	Test Cases	Bayes	TriBayes	Winnow	RuleS
accept, except	173	50	88.0	82.0	92.0	92.0
affect, effect	178	49	95.9	95.9	100	100
among, between	764	186	75.3	75.3	84.4	84.4
amount, number	460	123	82.9	82.9	86.2	87.8
begin, being	559	146	91.8	97.3	95.9	97.9
country, county	268	62	85.5	85.5	96.8	96.8
lead, led	173	49	79.6	83.7	93.9	93.9
passed, past	307	74	89.2	95.9	95.5	90.5
peace, piece	203	50	90.0	90.0	94.0	90.0
principal, principle	147	34	85.3	88.2	94.1	91.2
quiet, quite	264	66	89.4	95.5	90.9	93.9
raise, rise	98	39	74.4	76.9	89.7	84.6
than, then	2096	514	93.2	94.9	97.9	96.1
weather, whether	239	61	96.7	93.4	100	100