# Improving Word-Based Predictive Text Entry with Transformation-Based Learning

David J. Brooks and Mark G. Lee

School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
d.j.brooks, m.g.lee@cs.bham.ac.uk

**Abstract.** Predictive text interfaces allow for text entry on mobile phones, using only a 12-key numeric keypad. While current predictive text systems require only around 1 keystroke per character entered, there is ambiguity in the proposed words that must be resolved by the user. In word-based approaches to predictive text, a user enters a sequence of keystrokes and is presented with an ordered list of word proposals. The aim is to minimise the number of times a user has to cycle through incorrect proposals to reach their intended word.
This paper considers how contextual information can be incorporated into this prediction process, while remaining viable for current mobile phone technology. Our hypothesis is that Transformation-Based Learning is a natural choice for inducing predictive text systems. We show that such a system can: a) outperform current baselines; and b) correct common prediction errors such as "of" vs. "me" and "go" vs. "in".

## 1 Introduction

Since the widespread adoption of the Short Message Service (SMS) protocol, "text-messaging" has become a primary communication medium, and the need for efficient text-entry interfaces on mobile phone keypads has become paramount. The T9 – or "text on 9 keys" – interface allows users to enter alphabetic characters using only 9 keys. Each of these 9 keys is mapped to a set of alphabetic characters according to international standard ITU-T Recommendation E.161 [1, p.1], shown in Figure 1. Alternative mappings have been considered [2], but are unlikely to be adopted because some dialing systems require adherence to the ITU standard. Therefore, we consider only the ITU keypad mappings here.

The T9 mapping associates a single key with multiple letters, introducing ambiguity in key-entry, which must be resolved by the input interface. Early systems employed *multi-tap* interfaces, where each letter is unambiguously identified by a number of successive "taps" on a numeric key. More recently, multi-tap interfaces have been superceded by *single-tap* – or "predictive text" – interfaces[1], which have been shown to reduce the number of keystrokes required per character [3].

[1] In many single-tap interfaces, multi-tap entry is still used to enter novel words.


Advances in Natural Language Processing and Applications
Research in Computing Science 33, 2008, pp. 237-248

Received 31/10/07
Accepted 07/12/07
Final Version 17/01/08

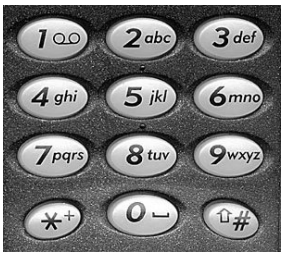


**Fig. 1.** A mobile phone keypad following the ITU-T E.161 mapping of numeric keys to alphabetic symbols. Non-standard mappings exist for key 1 (symbols such as punctuation), key 0 (spaces), and the * and # keys – one of which is usually assigned to "cycle"

In a single-tap interface, the ambiguity between characters is ignored upon entry: to enter a character requires only a single tap of the associated key. However, because there are multiple letters associated with the key-tap, the system must consider the possibility of extending the current word with each of the associated letters. Some of these extensions will be valid (part-)words, so are retained; others are invalid, so are discarded. Single-tap entry systems are surprisingly effective because, after the first few key-taps of a word, there are usually relatively few words matching that sequence of taps.

Despite improved performance, single-tap systems are still subject to ambiguity at the word level. For instance, the sequence of key-presses 4 – 6 – 6 – 3 corresponds to the words "good", "gone", "home", "hone", "hood" and "hoof", as well as representing the first four letters of words such as "immediate" and "honey". Single-tap entry systems are referred to as "predictive" because they must predict the user's intended word, given the sequence of keystrokes. The present work is concerned with how contextual information can be incorporated into this prediction process. Our hypothesis is that Transformation-Based Learning [4] is a natural choice for inducing effective predictive text systems, and has the advantage that the resulting systems can be implemented on current mobile phone technology.

The remainder of this paper is organised as follows. In Section 2 we give a formal description of word-based predictive text systems, and discuss some existing systems that achieve reasonable predictive behaviour. In Section 3 we discuss part-of-speech (POS) tagging. We argue that there are analogies between this task and word-based text prediction, and, with this in mind, we examine the Transformation-Based Learning (TBL) approach to POS tagging. In Section 4 we describe a novel instantiation of TBL for learning a word-based predictive text system. We describe a series of experiments in Section 5, including evaluations of our system against previous work. Finally, we discuss some limitations of the work in Section 6, and examine some possibilities for future research.

## 2 Word-Based Predictive Text Systems

In this paper, we consider word-based predictive text (henceforth WBPT) systems, where sequences of key-presses are mapped to sets of matching words. A WBPT system may be formally defined as follows.

Messages in a WBPT system are sequences of words, where words are delimited by space and punctuation characters. The system has a dictionary $D$ containing (a subset of) valid words from the text-entry language[2]. A user enters text by selecting words from the dictionary. Words are selected using a sequence of key-presses, where a key-press is represented by the corresponding numeral in the ITU-T E.161 mapping. A sequence of key-presses is represented as an ordered list of numerals, termed a *signature*. For example, the sequence of key-presses $4-6-6-3$ is represented by the signature $\langle 4663 \rangle$.

Each signature maps on to a (possibly empty) set of matching words – termed a *match-set* – which must be retrieved from the dictionary. This set is then ordered into a list, which we call the *match-list*. Match-list ordering is the principal difference between WBPT algorithms, and the most important factor influencing predictive performance. An optimum WBPT system should order the match-list from best to worst, according to context. Match-list ordering is central to our approach, and is used for both supervision and evaluation.

There are, of course, other ways to characterise predictive text systems. LetterWise [5] models transitional probabilities between letters, and so is amenable to novel compounding. However, this approach does not guarantee formation of valid words, and, more importantly, is incapable of resolving word-level prediction errors – such as deciding whether "good" or "home" better reflects a user's intended word. We believe that the approach described below resolves these issues, and may be readily extended to improve compounding systems.

### 2.1 Determining Match-List Order

The most naïve algorithms for ordering a match-list simply assign arbitrary orderings. This is clearly deficient, since, for a match-list of length $N$, an arbitrary ordering will start with the best candidate only $\frac{1}{N}$ of the time, while a worse match will be selected first $\frac{N-1}{N}$ of the time – meaning that users are highly likely to experience frustration in response to an arbitrary proposal!

Most existing WBPT systems employ a surprisingly effective algorithm: order the match-list according to descending word-frequency. We will call this the *most-frequent first* (MFF) algorithm. For exposition, we consider word frequencies taken from the 100-million-word British National Corpus (BNC)[3]. Table 1 shows the match-list for the signature $\langle 4663 \rangle$, ordered top-down.

MFF makes sensible predictions most of the time, but inevitably there are exceptions. A user will, at times, wish to words other than the most frequent, but

[2] Novel words may be added to the dictionary, but are ignored here because the associated cost is independent of the WBPT system.

[3] BNC statistics were obtained through the View/BNC interface [6].

**Table 1.** The match-set for $\langle 4663 \rangle$, with occurrence frequencies from the BNC.

| Frequency | Word |
|---|---|
| 80204 | good |
| 50539 | home |
| 18474 | gone |
| 960 | hood |
| 125 | hoof |
| 45 | hone |

since rarer words will be chosen less often (by definition) these exceptions are often tolerable. However, there are some predictions that are more problematic.

Consider the signature $\langle 63 \rangle$ and corresponding match-set $\{me,\ of,\ \ldots\}$. Both "of" and "me" are highly frequent words, occurring 2887937 and 2498656 times in the BNC respectively. MFF would use the match-list $(of, me, ...)$, and make at least 2498656 errors when classifying "me". These errors become more conspicuous in certain contexts. If the preceding word is "to" the situation changes dramatically: "to me" occurs 14635 times in the BNC, while "to of" occurs only 76 times[4]. MFF ignores this contextual information, predicting "of" every time.

### 2.2 Improving Baseline Performance by Considering Context

From the above example, we can see that MFF could be improved by incorporating contextual information – such as whether or not the previous word is "to". The HMS system [7] adopts this approach, supplementing a dictionary with a word bigram model. Each prediction is conditioned on the previous word, and HMS backs-off to MFF where reliable bigram statistics are unavailable. While this approach offers clear performance gains, bigram models have a large memory requirement – $O(n^2)$ where $n$ is the size of the dictionary.

Other recent approaches have employed "common-sense" reasoning [8], using ontological databases to create pragmatic links in order to aid prediction. However, we believe that most prediction errors can be resolved with local syntactic information, and that common-sense approaches deal with more peripheral cases. In addition, common-sense approaches require significant amounts of both memory and computation.

The operating environment for a predictive text system is typically a mobile device such as a cellular phone handset. While such devices have become increasingly powerful, the processing constraints are still relatively modest. Predictive text systems need to operate in real-time – at least as fast as rapid "texters" – and although systems like MFF are viable within these constraints, more expensive $n$-gram and reasoning techniques are not. Although these approaches may become viable in the long-term, we believe that practical improvements to WBPT may still be achieved in the short-term.

[4] Most occurrences of "to of" may be ascribed to mistranscription of "have" – as in "You ought to of seen his face" (found in document ACB of the BNC).

## 3 Supervised Part-of-Speech Tagging and Predictive Text

Part-of-speech (POS) tagging is the process of identifying the syntactic role of a word from its surface form and surrounding context. Tagging is non-trivial, as words may be ambiguous, often covering distinct "senses", possibly belonging to different parts-of-speech. For instance, "hope" can be either a noun or a verb.

In many ways, the scenario of POS tagging is similar to the word-based predictive text problem. First, simple taggers attempt to find the best tag for a given word, while we are attempting to find the correct word for a given signature. Second, training a supervised POS tagger requires a corpus where words are annotated with their correct tags. Third, an effective baseline system for supervised tagging is simply to assign the tag that accounts for most tokens of a given word type, which is equivalent to our MFF predictive text system. Finally, supervised tagging systems incorporate contextual information to improve upon the baseline system. Markov Model taggers (such as those described in [9]) condition the choice of the current tag on both the word and the $n$ previous tags – much as $n$-gram WBPT systems condition predictions based on previous words.

The above similarities suggest that POS tagging systems are analogous to WBPT systems, and effective POS tagging algorithms may yield improved WBPT systems. With this in mind, we will now discuss the Brill tagger – a rule-based POS tagger for incorporating contextual cues into the tagging process – which we believe can be readily adapted for WBPT.

### 3.1 Transformation-Based Learning

$n$-gram techniques haven proved successful for both tagging and WBPT, being capable of modelling local contextual information. However, $n$-grams devote considerable space to modelling rare events. Alternative learning strategies exist that incorporate local context while modelling primarily common events, and this idea is central to Transformation-Based Learning (TBL) [4]. TBL is an error-driven learning technique, which aims to improve a upon any given annotation system by identifying common errors and proposing rules to correct these errors where they occur. We will discuss the nature of a Transformation-Based learner in terms of POS tagging, with a view to treating WBPT within the same framework.

The typical operating environment for a TBL system is shown in Figure 2. A TBL system takes as input an initial guess at the correct POS for a corpus. Often, this is the output of a baseline (like MFF), but may be the output of any tagging system. The TBL learner then compares this initial guess to the "truth" – a gold-standard annotated version of the text, which in the case of tagging is manually-tagged corpus data. From this, the learner creates a list of errors, ordered by some objective criterion (usually just the frequency of the error, but any objective criterion will suffice).

The list of errors is used to propose correction rules, or *transformations*. Each transformation may be decomposed into two parts:

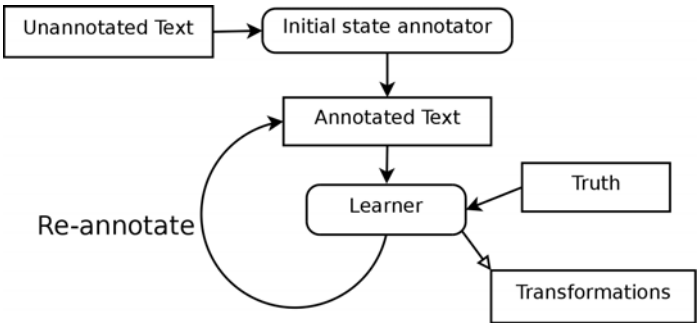


**Fig. 2.** The training environment for a Transformation-Based Learning system

**A rewrite rule** A rule that "transforms" an incorrect tag into the correct tag.
**A triggering environment** Some set of observed contextual features that indicate when a particular transformation should be used.

As an example, we might consider feasible transformations for "hope", which occurs in the BNC as both a verb (VVB - 7827 times) and as a noun (NN1 - 4106 times)[5]. Accordingly, the *most-frequent tag* system will always designate "hope" to be a verb. Clearly, from the frequencies, there are at least 4106 cases where this classification is incorrect, which we might correct using transformation. One conceivable transformation of "hope" might be:

| Rewrite rule: change tag from VVB to NN1 |
| --- |
| Trigger env.: previous word is "the". |

In the BNC, this transformation would correct 1490 mistagged instances of "hope".

TBL is a greedy algorithm. At each iteration, the best transformation – according to the objective criterion – is selected and applied to the corpus. This re-annotated corpus is then used as input to the next iteration. Learning continues until the level of improvement becomes negligible.

TBL has been shown to be a very effective algorithm for POS tagging, and this success is due to a number of factors. First, it incorporates contextual information only where it is useful for tagging decisions. Second, it is easy to train, although it does require a manually-tagged corpus as input. Finally, the resulting transformation rules are compact and more easily understandable than stochastic models. We believe that these features make Transformation-Based Learning ideal for WBPT, and will now consider how the learner must be altered for this new domain.

[5] The BNC contains 17192 occurrences of "hope", and VVB and NN1 are the two most frequent tags. Verb-infinitives, proper-nouns and compound tags (e.g. VVB-NN1) account for the remainder, but do not affect our example.

## 4 A TBL Approach to Predictive Text

In this section, we describe a novel instantiation of Transformation-Based Learning for WBPT. First, we describe our procedures for deriving a training corpus, discussing the form of data and nature of supervision. Second, we specify the form of transformations we use for WBPT. Third, we discuss a definition of error that is based on the human experience of single-tap interfaces. Finally, we describe a novel objective criterion for measuring prediction errors, incorporating this that is similar to the human experience of correction, and finally, considering how transformations should be specified in this domain.

### 4.1 Data and Supervision

As we observed earlier, TBL is primarily a supervised learning framework, and as such requires a training set that is annotated with correct prediction decisions. For WBPT, we require a data-set that shows the correct word for a given signature in the context of a message. Although such data-sets are not widely available, we can derive them automatically from any raw text corpus. Each word maps to a single signature, which can be trivially retrieved to create our "tagged" corpus. Consider the following sentence:

I want to go home.

For each word, we derive the corresponding signature and label the text:

4/I 9268/want 86/to 46/go 4663/home

From this we can also derive a signature-only version of this sentence:

4 9268 86 46 4663

This method can be used to generate a signature corpus, which can act as input to any WBPT system. The output of such a system will have the same sequences of signatures, but may assign different words according to the prediction model. For instance, MFF would produce the following prediction:

4/I 9268/want 86/to 46/in 4663/good

Thus, we have a gold-standard tagged text, and an input corpus comprising only of signatures. We can use the gold-standard as a training resource for supervised learning, and as a sample from which we can estimate signature-word frequencies for our baseline algorithm.

### 4.2 Transformations for Predictive Text

The form of transformations for predictive text is similar to that for POS tagging, with some important differences. Recall that transformations comprise a rewrite rule and a triggering environment. For WBPT, our rewrite rules alter the order of the match-list by promoting the correct word to the head of the list.

The triggering environments specify which contextual features are important for correcting erroneous predictions. There are two important differences between POS tagging and predictive text that affect triggering environments. First, a tagger may examine the components (e.g. morphemes) of the word being tagged, whereas in predictive text the mapping between signature and word precludes this. Second, text messages are entered in linear order, whereas POS taggers are conventionally applied to entire sentences. While the Brill tagger considers subsequent POS tags as context, this is not possible in a linear entry system. In our WBPT system, we limit triggering environments to the previous two words in the sentence.

### 4.3 Measuring Prediction Error

Once we have our gold-standard corpus, we need to be able to measure the prediction error of a given WBPT system. The human experience of correcting predictive text systems provides a natural way to do this. Upon entering a signature, a user is presented with a list of matching words. The user has to cycle through this list until they find their intended word, by pressing a designated *cycle key*[6]. Thus, where the correct word heads the list, no cycles are required; where the correct word is 4th in the list, 3 cycles are required. This quantity – which we will call the *cycle count* – naturally represents the error of any given word prediction.

We can trivially calculate cycle-count from a word and its match-list: the cycle-count is simply the index of that word within the match-list, where indices start from zero. The prediction error for a WBPT system over an entire corpus, $E$, is obtained by summing cycle-counts over all predictions.

### 4.4 Selecting Transformations

Transformation-Based Learning systems must be able to identify candidate transformations, from which the "best" selection is selected and applied at each learning iteration. Quality is traditionally defined in terms of prediction error, but in our scenario such a definition can lead to problems.

Recall the example signature $\langle 63 \rangle$, where MFF always predicts the word "of" but never predicts "me". Since "me" is a common word, this is a conspicuous errors and a good candidate for transformations. Now consider classifying $\langle 63 \rangle$ if we know that the previous word is "made" – the 15th most-frequent word preceding "me" in the BNC (1100 occurrences). Based purely on cycle-counts, "made" seems a strong contextual cue, and we might expect to learn the transformation:

| |
|---|
| Rewrite rule: change word from "of" to "me" |
| Trigger env.: previous word is "made". |

[6] The cycle key is not part of the ITU-T E.161 specification, but is usually mapped to either the * or # key.

which would correct 1100 misclassifications. However, "made" occurs *more frequently* before "of" (2783 occurrences in the BNC) – all of which would be erroneously transformed by the above transformation. Thus, the above transformation would do our system considerably more harm than good.

To alleviate this problem, we need a criterion that reflects the strength of relation between a rule and its triggering environment. Therefore, we define transformation quality as:

$$TQ = E_M^w \times |w| \times P(w|\sigma, T)^2 \quad (1)$$

where $E_M^w$ is the cycle-count for intended-word $w$ and match-list $M$, $\sigma$ is the signature of $w$ and $T$ is the triggering environment. The first term, $E_M^w$, is just the cycle-count as before, and when multiplied by the second term $|w|$, is the cycle-count across all occurrences of a word-type $w$ across the corpus. The third term $P(w|\sigma, T)$ represents the proportion of the time that $w$ is the correct word, given a signature and triggering environment[7].

We can examine this equation intuitively. The first two terms capture cycle-counts as before, making the system more likely to correct frequently misclassifications (e.g. "of" to "me"), than rare misclassifications (e.g. "good" to "hone").

The last term represents the *fidelity* of relationship between trigger and word. If a trigger is associated with many members of the match-list it will not help us make a correction. Thus, "to of" warrants a transformation because "to" occurs rarely with "of" but often with "me"; however "made of" does not warrant a transformation because "made" occurs frequently with both "of" and "me", and therefore affords little information with which to make a decision.

## 5 Experimental Setup

In the previous section, we described a novel TBL system for predictive text, which can be trained and evaluated on corpus data. In this section we will describe a series of experiments where we train and test our learning system and compare it to baseline and $n$-gram systems.

We trained our learning system on data taken from the BNC. Specifically, we created a single file containing all sentences of the BNC, and randomised the order between sentences (but not between words). From this corpus, we created 33 disjoint partitions, each comprising 180000 sentences (around 3.3 million words). We split each partition into a training set of 162000 sentences and a test set of 18000 held-out sentences – achieving a rough 90:10 split of training- to test-data in each partition.

We trained three systems on each of these corpora: the MFF baseline system; a bigram system with back-off; and our Transformation-Based learner, which

[7] We square $P(w|\sigma, T)$ only to increase our preference for strong relationships between triggering environments and predictions: large probabilities indicate strong relationships, small probabilities indicate weak relationships, and squaring a small probability will make it even smaller.

uses MFF as its initial input. Table 2 shows the mean and standard deviation in prediction accuracy for each system. We calculate significance levels ($p$-values) of differences in error-rates using a two-tailed paired $t$-test. The results show that

**Table 2.** Average accuracy (including standard-deviation) of three predictive text systems: MFF; a Bigram model with back-off; and Transformation Based Learning

| System | Accuracy | Stdev |
|---|---|---|
| MFF | 89.9189% | (0.0737) |
| Bigram with backoff | 90.5798% | (0.0739) |
| TBL | 90.6149% | (0.0736) |

MFF is a relatively good baseline system, performing at close to 89.92% accuracy. Both the bigram model and our TBL system achieve statistically significant increases in accuracy (of 0.66%, with negligible $p$-values) over the MFF baseline. This is to be expected as both these models incorporate contextual information, which should enhance prediction accuracy. Interestingly, our TBL system also achieves a small and significant increase in accuracy over the bigram model (0.04%, $p = 0.00027$). Initially this seems counter-intuitive because the bigram model encodes statistics even for rare preceding words. However, the results are understandable in terms of novel words in the held-out sample. To make a prediction for word $w_n$ where the preceding word $w_{n-1}$ was not part of the training data, the bigram model backs off to MFF and is therefore only as good as MFF over those cases. In contrast, our instantiation of TBL considers triggering environments covering either or both of two preceding words, $w_{n-2}$ and $w_{n-1}$. Thus, even where $w_{n-1}$ is novel, some accurate predictions can still be made if $w_{n-2}$ is part of a triggering environment. Such discontinuous affects are not captured in $n$-gram model.

The performance of our system is encouraging, so we should also examine our claims about memory requirements. For each experiment, around 1200 transformations were proposed – a negligible increase in comparison to the size of dictionary involved. We should also examine the transformations produced by the system, to see if they are only encoding important contextual information.

We chose one of the 33 experiments at random. During this experiment, our system proposed 1225 transformations, the top 10 of which are shown in Table 3. This list includes some of the very frequent errors identified earlier. The system successfully identifies transformations to correct "of" to "me", and "in" to "go", amongst others. Many of these transformations make syntactic sense: for instance, we are unlikely to see "to in", so the transformation "in" → "go": $\langle * \; to \rangle$ corrects this to "to go". However, we can also see some interesting exceptions. First, there are some contentious triggers: "there" → "these": $\langle * \; in \rangle$ represents the increased likelihood of "in these", but "in there" is a feasible construction. Second, there are some plausible constructions that would probably not be used much in text messages, such as "civil war". This reflects a bias specific to our

**Table 3.** The top 10 transformations for a randomly chosen experiment, ordered by number of corrections in training corpus. For each transformation, the most frequent triggers are shown. Each trigger is an ordered pair of preceding words, where $*$ represents any word

| # Corrections | Rule | Example Triggers |
|---|---|---|
| 3229 | them → then | $\langle * \; and \rangle$, $\langle * \; , \rangle$ , $\langle * \; . \rangle$, $\langle * \; is \rangle$ |
| 1930 | of → me | $\langle * \; to \rangle$, $\langle * \; for \rangle$, $\langle * \; told \rangle$, $\langle * \; let \rangle$ |
| 1613 | in → go | $\langle * \; to \rangle$, $\langle * \; n't \rangle$, $\langle * \; 'll \rangle$, $\langle * \; can \rangle$ |
| 1302 | on → no | $\langle there \; * \rangle$, $\langle award \; ref \rangle$, $\langle I \; have \rangle$, $\langle * \; oh \rangle$ |
| 1280 | there → these | $\langle * \; of \rangle$, $\langle * \; all \rangle$, $\langle * \; to \rangle$, $\langle * \; in \rangle$ |
| 1136 | up → us | $\langle * \; the \rangle$, $\langle * \; of \rangle$, $\langle * \; to \rangle$, $\langle * \; for \rangle$ |
| 957 | he → if | $\langle * \; even \rangle$, $\langle * \; see \rangle$, $\langle and \; , \rangle$, $\langle * \; asked \rangle$ |
| 655 | might → night | $\langle * \; last \rangle$, $\langle * \; the \rangle$, $\langle * \; a \rangle$, $\langle * \; all \rangle$ |
| 560 | an → am | $\langle * \; I \rangle$ |
| 556 | was → war | $\langle * \; the \rangle$, $\langle * \; world \rangle$, $\langle * \; civil \rangle$, $\langle * \; of \rangle$ |

training corpus – an issue we will discuss below. Finally, there are examples of ambiguity in the triggering environments: in the context $\langle * \; the \rangle$ "no" is an abbreviation of "number", and "us" is an abbreviation for "United States".

We should also examine the errors made by our original classifier, and the proportion of those errors that are corrected by our system. Figure 3 shows the 30 most costly words in our chosen test set. The bars show the original error

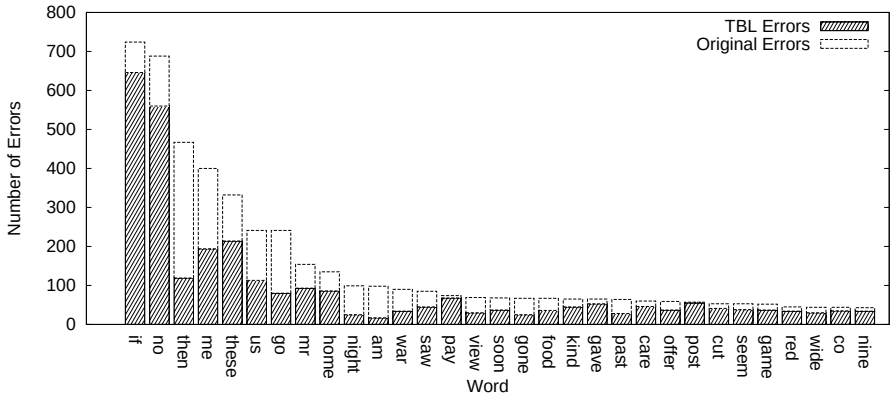


**Fig. 3.** 30 most-frequent errors made by MFF, with proportion remaining after TBL

count per word (in white), and the TBL error count as a proportion (hatched). For example, the first bar represents the word "if", for which there are 724 errors in MFF, and 646 in TBL system. Although the first two words – "if" and "no" – is largely uncorrected, for many of the subsequent words – including "then", "me", "these", "us", "go" and "am" – our system provides effective correction.

## 6 Future Work and Conclusions

We have seen that TBL can improve predictive text systems, but there are still areas where our approach could be improved. First, our system has been evaluated only in a Machine-Learning context. User-testing would give a better indication of performance *in situ*, and would permit comparative evaluation against other predictive text systems.

More crucially, our experiments use only BNC data, which does not adequately represent the language used in text messages. However, our methods are language-independent, and can easily be applied to more representative corpora and to other languages. Future work should consider training on SMS or instant-messaging corpora, and adapting our system to users' idiolects.

We have described a novel Transformation-Based learner for improving predictive text systems, in a language-independent manner. We have shown that the resulting WBPT systems achieve statistically significant improvements in prediction accuracy over bigram models, and that these systems are capable of correcting many problematic misclassifications in English. We believe our system offers tangible improvements for predictive text interfaces, and that it works within the constraints of current technology.

## References

1. International Telecommunication Union: E.161: Arrangement of digits, letters and symbols on telephones and other devices that can be used for gaining access to a telephone network. http://www.itu.int/rec/T-REC-E.161-200102-I/en (2001)
2. Gong, J., Tarasewich, P.: Alphabetically constrained keypad designs for text entry on mobile devices. In: CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM Press (2005) 211–220
3. MacKenzie, I.S.: KSPC (keystrokes per character) as a characteristic of text entry techniques. In: Mobile HCI '02: Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction, London, Springer-Verlag (2002) 195–210
4. Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. Computational Linguistics **21** (1995) 543–565
5. MacKenzie, I.S., Kober, H., Smith, D., Jones, T., Skepner, E.: Letterwise: Prefix-based disambiguation for mobile text input. Technical report, Eatoni (2001)
6. Davies, M.: The View/BNC interface. http://corpus.byu.edu/bnc/ (2004) Accessed on 23rd October 2007.
7. Hasselgren, J., Montnemery, E., Nugues, P., Svensson, M.: HMS: A predictive text entry method using bigrams. In: Proceedings of the Workshop on Language Modeling for Text Entry Methods, 10th Conference of the European Chapter of the Association of Computational Linguistics, Budapest, Hungary, Association for Computational Linguistics (2003) 43–49
8. Hawes, N., Kelleher, J.: Context-sensitive word selection for single-tap text entry. In: STAIRS 2004: Proceedings of the Second Starting AI Researchers' Symposium, IOS Press (2004) 217–222
9. Charniak, E., Hendrickson, C., Jacobson, N., Perkowitz, M.: Equations for part-of-speech tagging. In: AAAI. (1993) 784–789