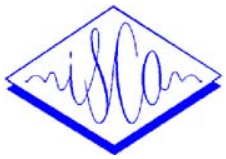
ISCA Archive
http://www.isca-speech.org/archive



INTERSPEECH 2004 - ICSLP
8th International Conference on Spoken Language Processing
ICC Jeju, Jeju Island, Korea
October 4-8, 2004

Transformation-Based Error Correction for Speech-to-Text Systems

Jochen Peters[1], *Christina Drexel*[2]

[1]Philips Research Laboratories, Weisshausstrasse 2, D-52066 Aachen
[2]Philips Speech Processing, Triesterstrasse 64, A-1101 Vienna
{jochen.peters,christina.drexel}@philips.com

10.21437/Interspeech.2004-554

Abstract

We present a universal approach to uncover and correct systematic local errors in complex speech-to-text systems. Whereas previous work to minimize speech recognition errors mostly relies on N-best lists or word lattices, our approach is merely based on the first-best system output. The paradigm of Transformation-Based Learning (TBL) is adapted from tagging-like applications to the more complicated task of text transformation which obstructs several basic TBL steps. On a professional spontaneous dictation task (including postprocessing and text formatting) we achieve error reductions of 9.6 %rel on held-out test data. A special benefit of the approach is the easy interpretation of the learned rules which may serve for diagnostic purposes.

1. Introduction

Automatic speech recognition (ASR) and natural language processing (NLP) techniques are nowadays used to convert spontaneous speech into formatted documents. Applications include the transcription of dictated reports and of meeting minutes. The full speech-to-text (STT) conversion becomes increasingly complex as more sophisticated NLP steps are included to postprocess the spontaneous input and to solve various formatting tasks. Unfortunately, all employed automatic techniques are error-prone and the complex system interactions may lead to errors which cannot be attributed to a single component. Error sources are thereby obscured and system tuning is handicapped.

This paper presents an approach to analyze and correct systematic errors of such integrated systems without tuning the individual system components. This is achieved in a Transformation-Based Learning like approach where we learn a sorted list of corrective rules which are executed on the first-best output of the STT system. Although we exploit statistical information from the error analysis while training the corrective system the final rules are fully deterministic. Experiments on real-life data from a professional transcription task with a tuned STT system show word error reductions by up to 9.6 %rel on held-out test data. The learned rules are easily interpreted and address a wide range of error types. Some of them hint at specific ASR or formatting problems and we may use the approach as a powerful diagnostic toolkit.

This paper is organized as follows: Sect. 2 reviews previous work on error minimization and Sect. 3 presents the basic paradigm of Transformation-Based Learning. This paradigm is adapted in Sect. 5 to the task of text correction which is depicted in Sect. 4. Experiments in Sect. 6 confirm the feasibility and success of our approach. Sect. 7 gives a summary and some outlook.

2. Previous work

Most ASR or NLP techniques follow the maximum-likelihood paradigm to deliver the most likely output for the given input. Systems, however, are normally evaluated in terms of word error rates and maximizing the likelihood of the complete output does not necessarily minimize the word error rate (WER). Much research has therefore focussed on explicit WER minimization of ASR systems. N-best list rescoring [4] and consensus decoding [6] exploit the probabilities assigned to a set of sentence hypotheses and select or construct a sentence with an expected minimum WER instead of the most likely sentence. Basically, consensus decoding converts a word lattice from ASR into a confusion network and selects, for each time interval, the most likely word. In [7] this approach was further improved using Transformation-Based Learning. Here, the decoder may select either the first or second most likely word per time interval, and the decision when to prefer the second-best word is conditioned on combinations of various features such as word identities, probabilities, the word duration etc.

Note that all these approaches work on pure ASR systems (without postprocessing) and exploit N-best lists or word lattices. The final sentence is always composed from word alternatives from these lists or lattices.

3. Transformation-Based Learning

Transformation-Based Learning (TBL) was originally developed for tagging tasks [1, 3] and has meanwhile been successfully applied to a variety of NLP problems including POS-tagging, prepositional phrase attachment, and parsing. Spelling correction [5] is another interesting application. Here, some typical word confusions (such as *than* $\leftrightarrow$ *then*) are predefined and conditions are learned for word-by-word replacements within any confusion set.

The basic principle of TBL is as follows:

1. We start with the output of an initial system (e.g. tagging, or text with spelling errors) and a reference version, i.e. the desired output (the "truth").
2. We define a set of allowable transformations, typically in the form of some rule templates.
3. Each allowed rule is scored by comparing some error rate (with respect to the reference output) of the current output and of the transformed output after applying the rule.
4. The best scoring rule is selected and applied to the current output. The so-transformed output is taken as new starting point and we iterate from step 3.

We thus learn a sorted list of rules which are applied in a cascaded fashion. The initial output O_0 is transformed by rule R_1, the result $O_1 = R_1(O_0)$ is further transformed by R_2 and so on. Note that each selected rule R_i is applied prior to re-evaluating all allowed rules whence all rule interactions are considered during training.

Note also that the scoring is very efficient for all tagging tasks (or for word-by-word replacements). Applying a rule to change one tag (or word) gives an error change of 0 or ± 1 since we have a one-to-one correspondence between each tag (or word) in O_i and its reference version.

4. Errors of Complex STT Systems

Complex speech-to-text applications include an initial ASR step to literally transcribe the spoken (spontaneous) input and several postprocessing steps. These may comprise the handling of disfluencies (hesitations, repetitions), automatic punctuation, text segmentation and insertion of section headings, and formatting of e.g. numbers, dates, acronyms etc. It is important to understand that the desired output is normally not a literal transcript but a well-formatted text. Considering this goal and all errors made by all automatic system components we thus face the following tasks:

- Correction of ASR errors (e.g. word confusions).
- Correction of spontaneous speech (removal of repetitions and redundant speech, insertion of skipped words, expansion of abbreviations etc.).
- Correction of formatting errors (which may be partly due to ASR errors).

5. Modified TBL for STT Error Correction

It is clear that we may encounter longer ranging error regions. Due to insertions and deletions and the formatting of multiple spoken words into one string (e.g. for dates) or vice versa (for acronyms) we loose any one-to-one correspondence between any word and its reference version.

We thus have to correct text regions of varying length (such as *third of May* $\rightarrow$ *05/03*). Following the idea of TBL we define the following basic rule template:

$$\textit{WrongText} \quad \rightarrow \quad \textit{CorrectText} \tag{1}$$

Many replacements need further conditions to prevent false applications. Additional templates include one or several "anchor" words $v, w, \ldots$ preceding or following the replacement:

$$v \quad \textit{WrongText} \quad \rightarrow \quad v \quad \textit{CorrectText} \tag{2}$$

$$\textit{WrongText} \quad w \quad \rightarrow \quad \textit{CorrectText} \quad w \tag{3}$$

$$v \quad \textit{WrongText} \quad w \quad \rightarrow \quad v \quad \textit{CorrectText} \quad w \tag{4}$$

This scheme may be further extended to more anchors including wildcards to ignore some nearby words. To generalize from individual anchor words we finally provide templates with word classes c as anchors such as

$$v \quad * \quad \textit{WrongText} \quad c \rightarrow \quad v \ \$1 \ \textit{CorrectText} \ \$2 \tag{5}$$

where \$1 and \$2 refer to the words matched by the wildcard $*$ and by the class c. An example of such a rule is

$$, \quad * \quad (\textit{and}|\textit{or}) \quad \rightarrow \quad , \ \$1 \ , \ \$2 \tag{6}$$

which inserts a missing comma in enumerations such as *red , blue or green* $\rightarrow$ *red , blue , or green.* Here, *WrongText* is empty, *CorrectText* provides the comma, and the rule uses a text format where punctuations are separate words (not attached to preceding words).

5.1. Rule generation from alignments

Our goal is to design rules which correct *observed* errors in some training data. The basic idea is thus to convert observed errors into a set of candidate rules following the above templates. We start with a standard Levenshtein alignment of the automatic STT output and some reference version of the texts. Consider the following example:

the	COR	*the*
patient	COR	*patient*
has	COR	*has*
	DEL	*a*
weird	SUB	*severe*
problem	COR	*problem*

Since word-to-word alignments are often ambiguous [1] we first extract complete error regions (SUB+INS+DEL) and convert them into the core replacement according to (1):

$$\textit{weird} \ \rightarrow \ \textit{a severe} \tag{7}$$

Then, all other templates are used to add conditional word or class anchors to this core rule. Each error region is thus converted into a set of more or less conditioned rules.

[1] *weird* might also be aligned with *a*

5.2. Scoring the rule candidates

An *exact* scoring of all rule candidates with the resulting WER change is prohibitive for large-scale applications with hundreds of thousands of rules and corpus sizes of several million words. Several *approximations* are conceivable: Local re-alignments after (locally) applying some rule accelerates the ΔWER estimation. Much more efficient, however, is an estimation of the error changes based on two simple counts: For each rule, we first check in how many cases the rule leads to an *exact repair* according to the alignments. In the above example, a rule like *weird problem* $\rightarrow$ *severe problem* would exactly repair one substitution error. (Note that we do not require that the full error region is repaired.) Second, we determine in how many cases the rule would transform a completely correct region (e.g. *the patient* $\rightarrow$ *the patients*). Each rule is thus characterized by three quantities: *GoodCnt* and *BadCnt* quantifying the exact repairs and the applications on correct text and *ErrLen* quantifying how many errors are repaired or introduced in either case. We now estimate the *expected error reduction*:

$$XER = ErrLen \cdot (GoodCnt - BadCnt) \qquad (8)$$

This simple estimate ignores all cases where the rule can be applied on (partly) incorrect text without exactly repairing the transformed region. Various control experiments where these cases were included in (8) with a positively or negatively weighted count showed that a weight close to zero gave both the best *XER* estimations and the best TBL performance whence we use (8) in all following experiments.

5.3. Rule selection

Although (8) allows to efficiently score all rule candidates we have to further relax the strict TBL paradigm where we would select *one* best-scoring rule, apply this single rule and re-generate and re-score all rule candidates. This, however, would require a re-alignment of the training data for each selected rule which is prohibitive.

We therefore decided to select many rules per iteration at the cost of ignoring some rule interactions. (Note that all rules are scored for the *current* system output.) We first sort all rules by *XER* and then discard all "overlapping" rules, i.e. we keep only the best-scoring rule from each set with the same core replacement *WrongText* $\rightarrow$ *CorrectText*. (This allows to select the best-performing anchors for any particular replacement.) Furthermore, if several rule candidates share the same left side and would thus map the same input to different outputs, we keep only the best-scoring rule. In any case, rules with $XER \leq 0$ are discarded. A lower threshold for *GoodCnt* allows to further remove rules with insufficient positive evidence.

5.4. Iteration

After selecting a set of rules we may iterate the procedure by first applying all rules in the determined order (highest *XER* first) and by re-aligning the transformed STT output with the reference texts. In each iteration, rule generation, scoring, and selection follow the above scheme.

As for the strict TBL paradigm, iterating the rule generation and selection allows to correct new or remaining errors from previous steps.

6. Experiments

6.1. Data

Our experiments are based on a real-life STT application for spontaneously dictated medical reports from various US hospitals. Each report was transcribed and formatted by an integrated STT system (ASR, automatic punctuation, stochastic formatting grammars) and manually corrected by experienced transcriptionists. Formatting includes numeric entities like date or blood pressure, the insertion or formatting of headings at section boundaries, and the layout of enumerated lists. As in (6) punctuations and formatting commands like <*newline*> are written as separate words. The database of 12 k reports (5.8 M words) was evenly split into TRAIN (80%), DEV (10%), and EVAL (10%). The overall WER is 35%.[2] Note that this is far above the plain ASR error rate for literal transcription due to necessary reformulations of the spontaneously spoken text, text rearrangements by the transcriptionists, and various formatting steps.

6.2. TBL setup

All experiments use the rule templates described in Sect. 5 with a maximum of two anchor words or classes[3] and limited to a trigram horizon (two words before and after each error region). Rules were generated from the TRAIN data. Rules which were seen only once were directly pruned prior to the scoring. To evaluate the reported robustness of TBL against overfitting we did separate experiments where the rule scoring was performed *either* on the TRAIN *or* on the independent DEV set. To analyze the generalization of rarely observed rules to the independent EVAL data we tested *GoodCnt* thresholds of up to 5.

6.3. Results and Discussion

Table 1 shows the results for an experimental setup where rules were generated and scored on TRAIN. Pruning by *GoodCnt* was disabled but we have $GoodCnt \geq 2$ since all rules which were seen only once during generation were discarded (Sect. 6.2). Here, the number of *selected* rules (#Rules in Table 1) ranges from 11% (it.1) to 2% (it.3) of the scored rule candidates.

[2] Error bars (95% confidence) are ± 0.2 %rel for TRAIN and ± 0.4 %rel for DEV and EVAL.

[3] with 7 handcrafted classes (headings, punctuations, ...)

Table 1: *Results with scoring on* TRAIN *(GoodCnt* ≥ 2*).*

It.	#Rules	CumΔWER [%rel]	
		TRAIN	EVAL
1	31425	-11.5	-7.9
2	9720	-13.9	-9.2
3	2728	-14.5	-9.6

From these selected rules, 15% (it.1) to 8% (it.3) are actually *not used* on the TRAIN data. This is due to the ignored rule interactions (Sect. 5.3) where the application of some rule destroys the match of some subsequently selected rule (due to transformed anchors or changes in *WrongText*).

Most learned rules are easily interpreted. We find:

- Text completion such as insertion of non-dictated articles (e.g. <*punctuation*> *patient* $\rightarrow$ \$1 *the patient*).
- Formatting aspects (e.g. insertion of colons in standard diagnoses like *lungs clear* $\rightarrow$ *lungs : clear*).
- Removal of repeated words (e.g. *the the* $\rightarrow$ *the*).
- Conversion of abbreviations and "dictation macros" which are typically expanded by the transcriptionists.
- Correction of systematic ASR errors including grammatical corrections (like *is been* $\rightarrow$ *has been*).

All control experiments resulted in deteriorations:

Discarding rules with *GoodCnt* $\leq N$ leads to a steady decrease of the achieved error reduction (in all 3 iterations) from 14.5 to 9.4 %rel on TRAIN and from 9.6 to 8.9 %rel on EVAL for $N = 1 \rightarrow 5$.

Scoring the candidates on DEV instead of TRAIN reduces the achieved error reduction on EVAL from 9.6 to 6.7 %rel (the error reduction on DEV is 14.1 %rel).

These findings indicate that TBL does *not* tend to overtraining which is in agreement with similar findings for tagging tasks [2]. We should thus exploit all available training data for both rule generation and scoring.

A final experiment emphasizes the importance of enough training data. We repeated the experiment from Table 1 with TRAIN replaced by DEV, thus reducing the learning data for rule generation and scoring by a factor of 8. This reduces the achieved error reduction on EVAL from 9.6 to 5.3 %rel (the error reduction on DEV is now 10.4 %rel).

We finally note that the simple *XER* estimate (8) performs reasonably well: comparing the actually achieved ΔWER per iteration (on the scoring data) with the accumulated *XER* of all used rules we find typical deviations by less than 10 % in all performed experiments (max. 25 % for very small ΔWER). For the first iteration of Table 1 we have a quasi perfect estimate (ΔWER was 0.2 % overestimated).

7. Conclusions and Future Work

The simple paradigm of TBL was adapted for the task of text transformation and our results demonstrate the feasibility and success of this approach. The technique is universal since we operate on the first-best output of the STT system and do not rely on predetermined confusion lists. An important practical advantage is the high diagnostic value of the learned rules. (Some of the uncovered error sources are meanwhile directly addressed in a new generation of our STT system.)

We should note that the approach is not restricted to speech-to-text systems since we learn typical corrections on some text (here, these are the editing steps of the transcriptionists on the automatic texts). We may thus envisage other applications such as, e.g., text normalization if some training texts are manually "standardized".

Future work should address a data-driven design of the used word-classes and the potential of *negative* conditions to prevent false applications. Tagging the STT output may provide additional features which can be exploited to condition the rules' application.

8. Acknowledgements

We are grateful to Evgeny Matusov (now at RWTH Aachen) for valuable discussions and initial experiments.

9. References

[1] Eric Brill. A Simple Rule-Based Part-of-Speech Tagger. *Proc. Third Conf. on Applied NLP, ACL*, 1992.

[2] Lance A. Ramshaw and Mitchell P. Marcus. Exploiting the statistical derivation of transformational rule sequences for part-of-speech tagging. *Proc. of the Balancing Act Workshop*, pp. 86–95, ACL, 1994.

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

[4] Andreas Stolcke, Yochai Konig, and Mitchel Weintraub. Explicit Word Error Minimization in N-Best List Rescoring. *Proc. Eurospeech*, vol. 1, pp. 163–165, 1997.

[5] Lidia Mangu and Eric Brill. Automatic Rule Aquisition for Spelling Correction. *Proc. ICML*, pp. 187–194, 1997.

[6] Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding Consensus Among Words: Lattice-Based Word Error Minimization. *Proc. Eurospeech*, vol. 1, pp. 495–498, 1999.

[7] Lidia Mangu and Mukund Padmanabhan. Error Corrective Mechanisms for Speech Recognition. *Proc. ICASSP*, vol. 1, pp. 29–32, 2001.