CYBERPUNC: A LIGHTWEIGHT PUNCTUATION ANNOTATION SYSTEM FOR SPEECH

Doug Beeferman   Adam Berger   John Lafferty

School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
dougb,aberger,lafferty@cs.cmu.edu

ABSTRACT

This paper describes a lightweight method for the automatic insertion of intra-sentence punctuation into text. Despite the intuition that pauses in an acoustic stream are a positive indicator for some types of punctuation, this work will demonstrate the feasibility of a system which relies solely on lexical information. Besides its potential role in a speech recognition system, such a system could serve equally well in non-speech applications such as automatic grammar correction in a word processor and parsing of spoken text. After describing the design of a punctuation-restoration system, which relies on a trigram language model and a straightforward application of the Viterbi algorithm, we summarize results, both quantitative and subjective, of the performance and behavior of a prototype system.

1. INTRODUCTION

The requirement that conventional speech dictation systems impose on the user to enunciate punctuation can often be an annoyance and in some situations even an impossibility. "Speaker-unaware" transcription systems, in which the speaker doesn't know (or care) that he is speaking to a dictation system, are one setting in which the enunciated-punctuation requirement cannot be enforced. For instance, a broadcast-news transcription system (Informedia [5], for example) cannot rely on the speakers to pronounce "comma" or "semicolon" wherever necessary in the transcribed text. Telephony and other remote speech applications (such as the answering system described in [6]) share the injunction against requiring the user to speak unnaturally. Ultimately, we imagine that all speech recognition systems will automatically insert punctuation as accurately and efficiently as a professional transcriptionist. The system we propose is, so far as we know, a first step in that direction.

Figure 1 depicts how a punctuation-restoration system could be incorporated into a conventional dictation system. The user's dictated speech is the input to the system; it passes through a conventional (punctuation-free) dictation system, which produces an N-best list, whose entries lack punctuation. A post-processing step then adds punctuation to each hypothesis by applying a trigram model "extended" from the baseline model to account for punctuation. A new N-best list, sorted by score assigned by the extended language model, is the final output.

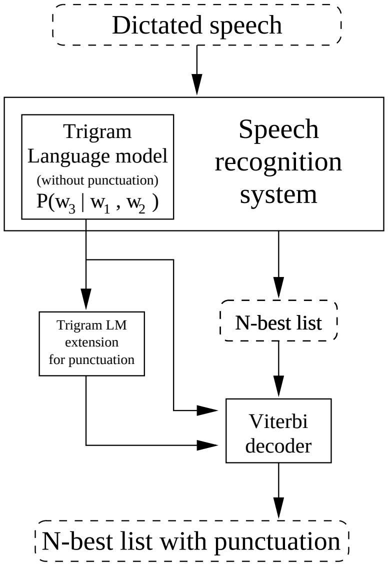


Figure 1: Automatic insertion of punctuation in a speech dictation system.

The paper will proceed as follows. After reviewing related work in Section 2, we describe the decoding algorithm in Section 3. A strict comparison of the output of a prototype system[1] against a reference corpus is of dubious value in settings such as these, where multiple correct answers exist. We address this issue by reporting in Section 4, alongside the standard statistical benchmarks of precision, recall and exact match, the results of human evaluation of the system's performance.

This work was partially funded by DARPA AASERT award DAAH04-95-1-0475, NSF grant IRI-9314969, and an IBM Cooperative Fellowship.

[1] An interactive demonstration of the system is online at http://www.link.cs.cmu.edu/cyberpunc

2. BACKGROUND

Punctuation has thus far received scant attention from the speech community, despite the obvious ergonomic advantage of a dictation system that frees the user from pronouncing it.

Though we are aware of no prior work on predicting punctuation in the lexical stream, variations of the phrase breaking task have received considerable attention. Noun phrase bracketers and "chunkers" [4, 2] are useful in applications that require some level of phrase-level segmentation but cannot afford the resources required by a full-blown parser. The intelligibility of a text-to-speech system, for example, depends partly upon its ability to assign convincing intonation to regions of the input sentence and to insert pauses where appropriate [3]. Phrase breaking algorithms typically use part-of-speech n-gram models or the Viterbi algorithm to predict the bracketing defined in a corpus of parse trees.

Symbol	Frequency
,	4.658%
.	4.174%
“ ”	1.398%
( )	0.211%
?	0.0389%
!	0.00520%

Table 1: Some common punctuation symbols in the 42 million token Wall Street Journal corpus, and their frequencies.

For the remainder of this paper we shall focus on the comma, the most frequent and unpredictable punctuation mark (Table 1). The comma exhibits a wide range of applications in text (Table 2), and disambiguating a comma's role in a sentence is a separate issue [7] from identification, the focus of this paper.

Other intra-sentence punctuation marks may be amenable to an approach similar to the one we present, although it can be argued that sentence-terminating punctuation such as question marks and exclamation marks require substantially more semantic insight [10].

3. DECODING

The comma-restoration algorithm relies on a punctuation-aware trigram language model $\mathcal{Q}$. Below we describe how such a model can be constructed from a trigram model $\mathcal{P}$ without punctuation, using minimal space.

The input to the comma-restoration algorithm is a sentence $\mathbf{x} = x_0 x_1 x_2 \ldots x_n$ lacking commas; the output is a sentence $\mathbf{y} = y_0 y_1 y_2 \ldots y_{n+c}$ containing the same words and $c \geq 0$ commas inserted in a subset of the $n-1$ intra-word positions of $\mathbf{x}$. We entertained two different decoding strategies, both of which consider as candidate decodings all 2^{n-1} possible hypotheses.

3.1. No insertion-penalty decoding

The first algorithm, which we denote by Algorithm A, calculates

$$y^\star = \mathrm{argmax}_{\mathbf{y}} \prod_{i=0}^{n+c} q(y_i \mid y_{i-2} y_{i-1}), \tag{1}$$

where $q(y_i \mid y_{i-2}y_{i-1})$ is the trigram model probability assigned by the model $\mathcal{Q}$ to word y_i following the bigram $y_{i-2}y_{i-1}$. Finding $y^\star$ can be accomplished with an application of the Viterbi algorithm; a sample lattice is depicted in (2b), with the transition probabilities given by the (punctuation-aware) trigram model probabilities of $\mathcal{Q}$. A rather more succinct perspective is provided by figure (2a), the finite-state machine governing the transitions in the trellis.

3.2. Insertion-penalty decoding

Algorithm B is similar to A except that not hypothesizing a comma in the i'th position now has a "cost" of $1 - q(, \mid y_i y_{i-1})$. That is, we seek

$$y^\star = \mathrm{argmax}_{\mathbf{y}} \prod_{i=0}^{n+c} q(y_i \mid y_{i-2} y_{i-1})(1 - q(, \mid y_{i-2} y_{i-1}))^{\delta(y_i)} \tag{2}$$

where

$$\delta(y_i) = \begin{cases} 0 & y_i = , \\ 1 & \text{otherwise} \end{cases}$$

In this approach, the score of each hypothesis is the product of $2n-2$ probabilities, whereas in Algorithm A, the score of a hypothesis with c commas is the product of $n-1+c$ probabilities. One could view this model as an HMM with two states and with output-dependent transition probabilities given by $\mathcal{Q}$.

3.3. Constructing a punctuation-aware LM

We have made the claim that the proposed system is lightweight. Yet it requires, in addition to the punctuation-free language model $\mathcal{P}$ already present in a speech dictation system, a second trigram model $\mathcal{Q}$, trained on data with punctuation. But given a trigram model $\mathcal{P}$, one can realize a model $\mathcal{Q}$ using a small number of additional parameters.

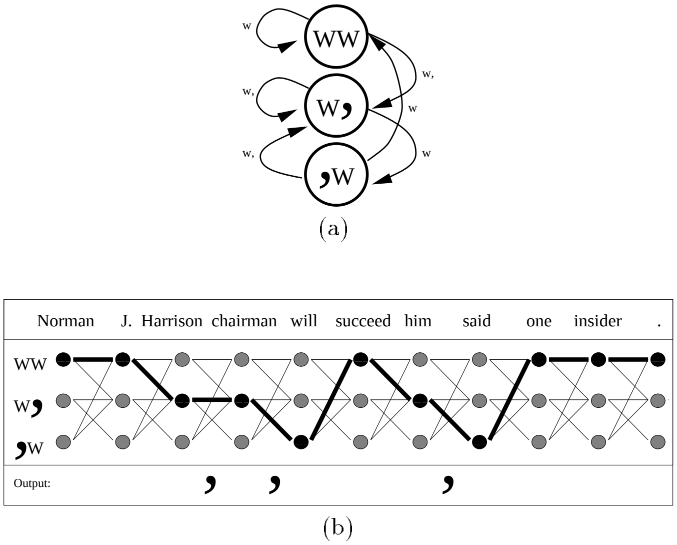
The method of extending $\mathcal{P}$ involves introducing a set of "corrective" parameters $\lambda(w_1, w_2)$, where $\lambda(w_1, w_2)$ is an estimate of the probability of a comma following the bigram (w_1, w_2). Just as the counts in a standard trigram model are sparse, so we can be sure that only a small number of parameters will be necessary to account for $\lambda(\cdot)$. For the experiments reported in section 4, for instance, we used a trigram model which contained $1,265,577$ trigrams; of these, only $185,420$ (14.7%) contained commas.

Defining $Z = p(y \mid y_{-2}y_{-1}) + \lambda(y_{-2}y_{-1})$, the probability of the extended model $\mathcal{Q}$ is given by

$$q(y \mid y_{-2}y_{-1}) = \begin{cases} Z^{-1}\lambda(y_{-2}y_{-1}) & y \text{ is a comma} \\ Z^{-1}p(y \mid y_{-2}y_{-1}) & \text{otherwise} \end{cases}$$

Textual role	Example
Around adverb phrases	At the time, Mr. Gatward said his friendship...
Between independent clauses	...keep an eye on the stock market, because "if the stock market rallies...
Around coordinate clauses	Institut Merieux, which already holds an eleven percent stake, is...
Enclosing appositives	Roger Sutton, executive vice president, was appointed...
Between elements in a series	While claiming that penalties, legal fees and interest have driven the value...
Before and after quotation marks	..."Maybe Akzo can surprise the investment world a bit," said Jaap Visker...
Within dates	...postmarked no later than Sunday, Oct. 7, and...
Within places	...a vaccine and bioresearch firm based in Lyon, France, is...

Table 2: The most frequent roles of the comma in the Penn Treebank corpus.



(a)

(b)

Figure 2: (a) The finite state machine underlying the three-state Viterbi decoder for the comma symbol; (b) The lattice for a sample sentence, with the Viterbi path highlighted.

4. EVALUATION

To evaluate our systems empirically, we selected the Penn Treebank corpus [8] of sentences from the Wall Street Journal. We used the standard 2317-sentence test suite stripped of all punctuation. We ran the decoding algorithms described above to produce punctuated output that was then compared with the original test set. The results for our algorithms and for the strategy of never hypothesizing a comma are summarized in Table 4.

As expected, decoding with a gap insertion penalty (Algorithm B) increased precision but lowered recall over Algorithm A. This can be explained by B's relative conservatism in hypothesizing a comma. Although it has poorer F-measure performance, Algorithm B nonetheless achieved higher sentence (exact match) accuracy. We feel that sentence accuracy is more indicative of real-world utility, since there is some overhead associated with correcting system errors if any mistakes are made at all.

4.1. Qualitative evaluation

The most valuable qualitative metric of success for a speech recognition system is user satisfaction. Accordingly, we would like to gauge how satisfying the punctuation choices of our system are. (Whether grammatical rules and conventions leave no ambiguity in comma placement is beside the point. The acceptability of our system hinges on whether its decisions align with the user's tastes.)

We applied Algorithm B to 100 sentences drawn randomly from the test set. The resulting machine-punctuated sentences where randomly intermingled with the original 100 sentences with the punctuation as they appeared in the Treebank. We then asked human judges to label each sentence in this 200-sentence list as acceptable or not with respect to comma placement.

Six judges of varied background participated in the evaluation. The results, summarized in Table 4.1, indicate that the performance of our system is qualitatively higher than the sentence accuracy rate would indicate.

5. CONCLUSIONS

We have presented a lightweight method for the automatic insertion of intra-sentence punctuation into text which relies exclusively on lexical information. Despite being quite straightforward in design, the system performs surprisingly well in most circumstances. Not surprisingly, the system performs poorly when the presence or absence of punctuation requires long-range lexical information to resolve. For instance, the prototype system correctly identifies the first comma in the following sentence, but not the second: Brazil similar to Mexico and South Korea is expected to negotiate.

Resolving such ambiguities lies beyond the scope of a second-order Markov model, and requires a model with richer conditioning information such as a trigger language model, described in [1]. Higher-order information—such as the parts of speech assigned by a tagger or even the structure of a parse tree hypothesized by a statistical parser—could also provide such long-range information, but at the expense of a less lightweight system. We are currently developing more sophisticated punctuation restoration schemes along these lines.

Algorithm	Sentence accuracy	Precision	Recall	F-measure	Token accuracy
A	53.3%	75.6%	65.6%	70.2%	96.6%
B	54.0%	78.4%	62.4%	69.4%	96.6%
none	32.9%	100.0%	0.0%	0.0%	93.8%

Table 3: Performance of three comma placement algorithms on the Penn Treebank test data. Algorithm A is the Viterbi 3-state decoder without the insertion penalization described in Section 3. Algorithm B is the 3-state decoder with insertion penalization. The "none" algorithm hypothesizes no commas at all. F-measure is a weighted combination of precision P and recall R, $\frac{2PR}{P+R}$.

	Decoder/reference agree		Decoder/reference disagree		Overall	
	right	wrong	right	wrong	right	wrong
Reference	289	35	226	50	86%	14%
Decoder	289	35	107	169	66%	34%

Table 4: Results of human evaluations. Six human judges were asked to evaluate a set of 200 sentences, 100 of which were a machine punctuated versions of the original sentences from the treebank corpus. The total counts are given for the evaluations, divided into those sentences for which the corpus and machine punctuation agreed, and the set where they disagreed.

6. REFERENCES

[1] D. Beeferman, A. Berger, and J. Lafferty (1997) A model of lexical attraction and repulsion. Proceedings of the ACL-EACL Joint Conference

[2] A. Black and P. Taylor (1997) Assigning phrase breaks from part-of-speech sequences. Eurospeech '97

[3] N. Campbell, T. Herbert, and Ezra Black (1997) Parsers, prominence, and pauses. Eurospeech '97

[4] K.H. Chen and H-H. Chen (1994) Extracting Noun Phrases from Large-Scale Texts: A Hybrid Approach and its Automatic Evaluation. Proc. ACL

[5] M. Christel, T. Kanade, M. Mauldin, R. Reddy, M. Sirbu, S. Stevens, H. Wactlar (1995). Informedia Digital Video Library. Comm. of the ACM 38:4, 57–58.

[6] B. M. Lobanov, S. V. Brickle, A. V. Kubashin, T. V. Levovskaja (1997). An intelligent telephone answering system using speech recognition Eurospeech '97

[7] M. McAllister (1989). The problems of punctuation ambiguity in fully automatic text-to-speech conversion Eurospeech '89

[8] M. Marcus and B. Santorini and M. Marcinkiewicz (1993). Building a large annotated corpus of English: the Penn Treebank, Computational Linguistics 19:1.

[9] G. Nunberg (1990) The linguistics of punctuation. CSLI Lecture Notes No. 18, University of Chicago Press

[10] J. Reynar and A. Ratnaparkhi (1997). A maximum-entropy approach to identifying sentence boundaries. Proceedings of the Fifth Conference on Applied Natural Language Processing.

[11] M. White (1995) Presenting punctuation. Proceedings of the Fifth European Workshop on Natural Language Generation Leiden, The Netherlands, pp 107–125

7. SAMPLE OUTPUT

The Viterbi decoder produces, in most cases, quite reasonable output. Below we provide a randomly interleaved collection of sentences blindly selected from the decoder output and from the Treebank itself. The reader is invited to distinguish between treebank (R) and decoder (D) sentences; the results are provided in a footnote[2]

1. And although Pennsylvania and Massachusetts suffered slight declines earlier in the decade, they are growing again.

2. China, which had been putting in huge orders for polyethylene, abruptly halted them.

3. CNN recently gave most employees raises of as much as 15%, but they're still drastically underpaid compared with the networks.

4. Meanwhile, during the S&P trading halt S&P futures sell orders began piling up, while stocks in New York kept falling sharply.

5. At the time, Hertz said its annual fees to those airlines amounted to $20 million.

6. Japan, the EC Brazil, Mexico and South Korea provide about 80% of the steel imported to the U.S. under the quota program.

7. The account had billed about $6 million in 1988, according to leading national advertisers.

8. Many interpret the delay as an indication that regulators are skeptical about the proposal.

9. In the hands of a zealot like Lenny Bruce this double-edged blade could cut both the self and the audience to ribbons.

10. Do not say the TV sitcom, because that happens to be a genre that, in its desperate need to attract everybody and offend nobody, resembles politics more than it does comedy.

[2] 1. D, 2. R, 3. R, 4. D, 5. D, 6. D, 7. R, 8. R, 9. D, 10. R.