Intra-sentence Punctuation Insertion in Natural Language Generation

Zhu ZHANG[†], Michael GAMON[‡], Simon CORSTON-OLIVER[‡], Eric RINGGER[‡]

[†]School of Information
University of Michigan
Ann Arbor, MI 48109
zhuzhang@umich.edu

[‡]Microsoft Research
One Microsoft Way
Redmond, WA 98052
{mgamon, simonco, ringger}@microsoft.com

30 May 2002

Technical Report
MSR-TR-2002-58

Microsoft Research
One Microsoft Way
Redmond WA 98052
USA

Intra-sentence Punctuation Insertion in Natural Language Generation

Zhu ZHANG[†], Michael GAMON[‡], Simon CORSTON-OLIVER[‡], Eric RINGGER[‡]

[†]School of Information
University of Michigan
Ann Arbor, MI 48109
zhuzhang@umich.edu

[‡]Microsoft Research
One Microsoft Way
Redmond, WA 98052
{mgamon, simonco, ringger}@microsoft.com

Abstract

We describe a punctuation insertion model used in the sentence realization module of a natural language generation system for English and German. The model is based on a decision tree classifier that uses linguistically sophisticated features. The classifier outperforms a word n-gram model trained on the same data.

1 Introduction

Punctuation insertion is an important step in formatting natural language output. Correct formatting aids the reader in recovering the intended semantics, whereas poorly applied formatting might suggest incorrect interpretations or lead to increased comprehension time on the part of human readers.

In this paper we describe the intra-sentence punctuation insertion module of Amalgam (Corston-Oliver *et al.* 2002, Gamon *et al*. 2002), a sentence-realization system primarily composed of machine-learned modules. Amalgam's input is a logical form graph. Through a series of linguistically-informed steps that perform such operations as assignment of morphological case, extraposition, ordering, and aggregation, Amalgam transforms this logical form graph into a syntactic tree from which the output sentence can be trivially read off. The intra-sentence punctuation insertion module described here applies as the final stage before the sentence is read off.

In the data that we examine, intra-sentential punctuation other than the comma is rare. In one random sample of 30,000 sentences drawn from our training data set there were 15,545 commas, but only 46 em-dashes, 26 semi-colons and 177 colons. Therefore, for this discussion we focus on the prediction of the comma symbol.

The logical form input for Amalgam sentence realization can already contain commas in two limited contexts. The first context involves commas used inside tokens, e.g., the radix point in German, as in example (1), or as the delimiter of thousands in English, as in example (2).

(1) Ich habe DM 4,50.
"I have 4.50DM."
(2) I have $1,000 dollars.

The second context involves commas that separate coordinated elements, e.g., in the sentence "I saw Mary, John and Sue". These commas are treated as functionally equivalent to lexical conjunctions, and are therefore inserted by the lexical selection process that constructs the input logical form.

The evaluation reported below excludes conjunctive commas and commas used inside tokens. We model the placement of other commas, including commas that indicate apposition (3), commas that precede or follow subordinate clauses (4) and commas that offset preposed material (5).

(3) Colin Powell, the Secretary of State, said today that…
(4) After he ate dinner, John watched TV.
(5) At 9am, Mary started work.

2 Related work

Beeferman *et al.* (1998) use a hidden Markov model based solely on lexical information to predict comma insertion in text emitted by a speech recognition module. They note the

difficulties encountered by such an approach when long distance dependencies are important in making punctuation decisions, and propose the use of richer information such as part of speech tags and syntactic constituency.
The punctuation insertion module presented here makes extensive use of features drawn from a syntactic tree such as constituent weight, part of speech, and constituent type of a node, its children, its siblings and its parent.

3 Corpora

For the experiments presented here we use technical help files and manuals. The data contain aligned sentence pairs in German and English. The alignment of the data is not exploited during training or evaluation; it merely helps to ensure comparability of results across languages. The training set for each language contains approximately 100,000 sentences, from which approximately one million cases are extracted. Cases correspond to possible places between tokens where punctuation insertion decisions must be made. The test data for each language contains cases drawn from a separate set of 10,000 sentences.

4 Evaluation metrics

Following Beeferman *et al.* (1998), we measure performance at two different levels. At the token level, we use the following evaluation metrics:

- *Comma Precision:* The number of correctly predicted commas divided by the total number of predicted commas.
- *Comma Recall.* The number of correctly predicted commas divided by the total number of commas in the reference corpus.
- *Comma F-measure.* The harmonic mean of comma precision and comma recall, assigning equal weight to each.
- *Token accuracy:* The number of correct token predictions divided by the total number of tokens. The baseline is the same ratio when the default prediction, namely do not insert punctuation, is assumed everywhere.

At the sentence level, we measure *sentence accuracy,* which is the number of sentences containing only correct token predictions divided by the total number of sentences. This is based on the observation that what matters most in human intelligibility judgments is the distinction between correct and incorrect sentences, so that the number of overall correct sentences gives a good indication of the overall accuracy of punctuation insertion. The baseline is the same ratio when the default prediction (do not insert punctuation) is assumed everywhere.

5 Punctuation learning in Amalgam

5.1 Modeling

We build decision trees using the WinMine toolkit (Chickering, n.d.). Punctuation conventions tend to be formulated as “insert punctuation mark X before/after Y” (e.g., for a partial specification of the prescriptive punctuation conventions of German, see Duden 2000), but not as “insert punctuation mark X between Y and Z”. Therefore, at training time, we build one decision tree classifier to predict preceding punctuation and a separate decision tree to predict following punctuation. The decision trees output a binary classification, “COMMA” or “NULL”.
We used a total of twenty-three features for the decision tree classifiers. All twenty-three features were selected as predictive by the decision tree algorithm. The features are given here. Note that for the sake of brevity, similar features have been grouped under a single list number.

1. Syntactic label of the node and its parent
2. Part of speech of the node and its parent
3. Semantic role of the node
4. Syntactic label of the largest immediately following and preceding non-terminal nodes
5. Syntactic label of the smallest immediately following and preceding non-terminal node
6. Syntactic label of the top right edge and the top left edge of the node under consideration
7. Syntactic label of the rightmost and leftmost daughter node
8. Location of node: at the right edge of the parent, at the left edge of the parent or

neither

9. Length of node in tokens and characters
10. Distance to the end of the sentence in tokens and characters
11. Distance to the beginning of the sentence in tokens and in characters
12. Length of sentence in tokens and characters

The resulting decision trees are fairly complex. Table 1 shows the number of binary branching nodes for each of the two decision tree models for both English and German. The complexity of these decision trees validates the data-driven approach, and makes clear how daunting it would be to attempt to account for the facts of comma insertion in a declarative framework.

Model	**English**	**German**
Preceding punctuation	563	743
Following punctuation	664	633

Table 1 Complexity of the decision tree models in Amalgam

At generation time, a simple algorithm is used to decide where to insert punctuation marks. Pseudo-code for the algorithm is presented in Figure 1.

```
For each insertion point I
    For each constituent whose right boundary
    occurs at the token preceding I
            If p(COMMA) > 0.5
                    Insert comma
                    Do next insertion point
            End if
    End for each
    For each constituent whose left boundary
    occurs at the token following I
            If p(COMMA) > 0.5
                    Insert comma
                    Do next insertion point
            End if
    End for each
End for each
```

Figure 1 Pseudo-code for the insertion algorithm

The threshold 0.5 is a natural consequence of the binary target feature: p(COMMA)>p(NULL) implies p(COMMA)>0.5.

Consider the application of the Amalgam punctuation insertion module for one possible insertion point in a simple German sentence *Er las ein Buch das kürzlich erschien* “He read a book which came out recently”. The parse tree for the sentence is shown in Figure 2.

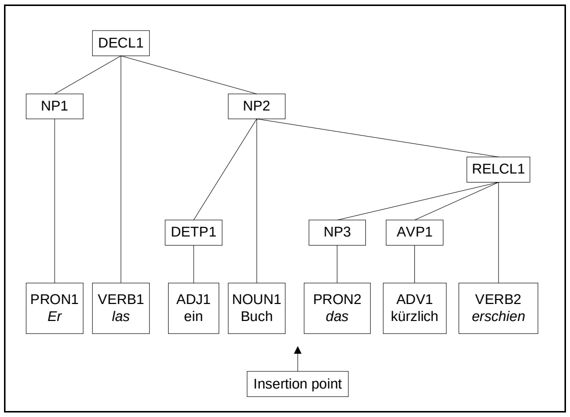

Figure 2 German parse tree

The scenario illustrated in Figure 2 is relatively straightforward. According to German punctuation conventions, all relative clauses, whether restrictive or non-restrictive, should be preceded by a comma, i.e., the relevant insertion point is between the noun *Buch* “book” and the relative clause *das kürzlich erschien* “which came out recently.”

When considering the insertion of a comma at the marked insertion point, Amalgam examines all constituents whose rightmost element is the token preceding the insertion point, in this case the noun *Buch* “book”. There is no non-terminal constituent whose rightmost element is the token *Buch*. The decision tree classifier for following punctuation is therefore not invoked.

Amalgam next considers all constituents whose leftmost element is the token to the right of the insertion point, in this case *das* “which”.[1] The constituents to be examined are NP3 (the projection of the pronoun), and RELCL1, the clause in which NP3 is the subject.

Consulting the decision tree for preceding punctuation for the node NP3, we obtain p(COMMA) = 0.0001. Amalgam proceeds to the next highest constituent, RELCL1. Consulting

[1] Note that many relative pronouns in German are homographic with determiners, a notable difficulty for German parsing.

the decision tree for preceding punctuation for RELCL1 yields p(COMMA) = 0.9873. The actual path through the decision tree for preceding punctuation when considering RELCL1 is illustrated in Figure 3. Because the probability is greater than 0.5, we insert a comma at the insertion point.

Label of top left edge is not RELCL and
Label is RELCL and
Part of speech of the parent is not Verb and
Label of rightmost daughter is not AUXP and
Label of leftmost daughter is not PP and
Label of smallest following non-terminal node is not NP and
Part of speech of the parent is Noun and
Label of largest preceding non-terminal node is not PP and
Label of smallest following non-terminal node is not AUXP and
Distance to sentence end in tokens is < 2.97 and
Label of top right edge is not PP and
Distance to sentence end in token is < 0.0967

Figure 3 Example of the path through the decision tree for preceding punctuation

5.2 Evaluation

In Table 2 we present the results for the Amalgam punctuation approach for both English and German.

	English	**German**
Comma recall	67.64%	87.54%
Comma precision	74.44%	85.44%
Comma F-measure	70.88%	86.47%
Token accuracy	98.02%	98.72%
Baseline accuracy	96.44%	95.35%
Sentence accuracy	76.90%	84.00%
Baseline accuracy	56.44%	47.30%

Table 2 Experimental results for comma insertion in Amalgam

Amalgam's punctuation insertion dramatically outperforms the baseline for both German and English. Interestingly, however, Amalgam yields much better results for German than it does for English. This accords with our pre-theoretical intuition that the use of the comma is more strongly prescribed in German than in English. Duden (2000), for example, devotes twenty-seven rules to the appropriate use of the comma. By way of contrast, Quirk *et al.*. (1985), a comparable reference work for English, devotes only four brief rules to the topic of the placement of the comma, with passing comments throughout the rest of the volume noting minor dialectal differences in punctuation conventions.

6 Language modeling approach to punctuation insertion

6.1 Modeling

We employ the SRI language modeling toolkit (SRILM, 2002) to implement an n-gram language model for comma insertion. We train a punctuation-aware trigram language model by including the comma token in the vocabulary. No parameters of the SRILM toolkit are altered, including the default Good-Turing discounting algorithm for smoothing.

The task of inserting commas is accomplished by tagging hidden events (COMMA or NULL) at insertion points between word tokens. The most likely tagged sequence, including COMMA or NULL at each potential insertion point, consistent with the given word sequence is found according to the trigram language model.

6.2 Evaluation

The results of using the language modeling approach to comma insertion are presented in Table 3.

	English	**German**
Comma recall	62.36%	74.62%
Comma precision	78.20%	89.64%
Comma F-measure	69.39%	81.45%
Token accuracy	98.08%	98.40%
Baseline accuracy	96.51%	95.30%
Sentence accuracy	74.94%	78.56%
Baseline accuracy	56.35%	47.26%

Table 3 Experimental results for the language modeling approach to comma insertion[2]

[2] Note that the baseline accuracies in Table 2 and Table 3 differ by a small margin. Resource constraints during the preparation of the Amalgam test logical forms led to the omission of sentences containing a total of 18 commas for English and 47 commas for

As Table 3 shows, the language modeling approach to punctuation insertion also dramatically beats the baseline. As with the Amalgam approach, the algorithm performs much better on German data than on English data.

Note that Beeferman *et al.* (1998) perform comma insertion on the output of a speech recognition module which contains no punctuation. As an additional point of comparison, we removed all punctuation from the technical corpus. The results were marginally worse than those reported here for the data containing other punctuation in Table 3. We surmise that for the data containing other punctuation, the other punctuation provided additional context useful for predicting commas.

7 Discussion and Conclusions

We have shown that for all of the metrics except comma precision the Amalgam approach to comma insertion, using decision trees built from linguistically sophisticated features, outperforms the n-gram language modeling approach that uses only lexical features in the left context. This is not surprising, since the guidelines for punctuation insertion in both languages tend to be formulated relative to syntactic constituency. It is difficult to capture this level of abstraction in the n-gram language modeling approach. Further evidence for the utility of features concerning syntactic constituency comes from the fact that the decision tree classifiers do in fact select such features (section 5.1). The use of high-level syntactic features enables a degree of abstraction over lexical classes that is hard to achieve with simple word n-grams.

Both approaches to comma insertion perform better on German than they do on English. Since German has a richer repertoire of inflections, a less rigid constituent order, and more frequent compounding than English, one might expect the German data to give rise to less predictive n-gram models, given the same number of sentences. Table shows the vocabulary sizes of the training data and the perplexities of the test data, with respect to the statistical language models for each language. Despite this, the n-gram language model approach to comma insertion performs better for German than for English. This is further evidence of the regularity of German comma placement discussed in Section 5.2.

Language	Vocab. Size	Perplexity
English	42084	96.128
German	84770	145.352

Table 4 Vocabulary size and perplexity for English and German

One advantage that the Amalgam approach has over the n-gram language modeling approach is its usage of the right context. As a possible extension of the work presented here and that of Beeferman *et al.* (1998), one could build a right-to-left word n-gram model to augment the left-to-right n-gram model. Conversely, the language model captures idiosyncratic lexical behavior that could also be modeled by the addition of lexical features in the decision tree feature set.

References

Beeferman D., Berger A. and Lafferty J. (1998) *Cyberpunc: A lightweight punctuation annotation system for speech.* IEEE Conference on Acoustics, Speech and Signal Processing. Seattle, WA, USA.

Chickering, D. Max. n.d. WinMine Toolkit Home Page. http://research.microsoft.com/~dmax/ WinMine/Tooldoc.htm.

Corston-Oliver, S., M. Gamon, E. Ringger, R. Moore. (2002) “An overview of Amalgam: A machine-learned generation module”. In review.

Duden. (2000) *Die deutsche Rechtschreibung.* Duden-Verlag: Mannheim, Leipzig, Wien, Zürich.

Gamon, M., S. Corston-Oliver, E. Ringger, R. Moore (2002) “Machine-learned contexts for linguistic operations in German sentence realization”. To be presented at ACL 2002.

Quirk, R., S. Greenbaum, G. Leech and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language.* Longman: London and New York.

SRILM. (2002) SRILM Toolkit Home Page. http://www.speech.sri.com/projects/srilm.

German.