Punctuation In The Bravice English-To-Japanese Machine Translation System

Jan Fornell
Sun Heights Nagashima B102
Daidoo 1-23-15, Kanazawa-ku
Yokohama, 236 Japan
Phone/fax: +81 45 781 6325
`PXQ00421@niftyserve.or.jp`

Abstract

There appear to be few large NLP systems that make any significant syntactic use of punctuation. This paper describes the English side of a sadly neglected full-scale machine translation system, developed 1987–91, which treated punctuation similarly to other lexical and phrasal categories, from morphology to the sentence, in a unificational framework. The treatment of most well-known (and some little-known) punctuation phenomena is discussed, together with the advantages and limitations of this integrated approach. I also give a brief overview of Japanese punctuation and discuss the general possibility and desirability of translating punctuation.

1 Background

The machine translation system described in this paper was developed jointly by Bravice International Inc., Tokyo, and Executive Computer Systems (ECS), Provo, Utah, from around early 1987. It is fully unificational from the dictionary on up, based on LFG, and was intended to be automatic (as opposed to user-assisted) for general text, with no or as little pre- and post-processing as possible. The output of the chart parser is a functional structure for the input sentence which is transferred to a corresponding functional structure for the target language, and then (with a bit of luck) a target sentence is generated to match. Although the primary focus (at least for Bravice) was on an English-to-Japanese system, the English and Japanese modules were completely separate and could be used with other languages as well, both for parsing and generation (in fact, English-Chinese and English-Korean/Korean-English systems were developed by ECS, using largely the same English component as the English-Japanese system). The linguistic rules are written in a custom high-level language which allows ID/LP format, optionality, Kleene star, both top-down and bottom-up rules, etc., but not variables over categories.

Of central importance is that the system was designed to run on "cheap" PCs,[1] rather than on expensive workstations like virtually all other commercial systems at the time. This of course put severe but interesting constraints on the system.

When I joined Bravice in the summer of 1987, there was already a small, functioning version of the English parser, which I believe was originally developed by ECS for their English-Chinese system. Over the next two years, I worked closely together with Joseph Pentheroudakis of ECS to expand those early English rules by several magnitudes. At the same time, a Japanese module and English, Japanese and English-Japanese dictionaries were developed and added to make a full-sized working system.

In late 1989, the relationship between the two companies broke down, and I continued working alone on the English side of the system, tuning the rules and the dictionary against raw text data and implementing an intricate constraint-passing framework, until Bravice suddenly went bankrupt in January 1991. Since then, the system has been in suspended animation, but is not strictly speaking quite dead yet, which is why most of this paper is in the present tense.

Due to unfortunate company policies, none of this work was ever presented in any papers or at any international conferences, and so remains virtually unknown in the computational linguistic community. This presentation of one aspect of the system is long overdue. However, it must be pointed out that since the system was never put on the market, there was never any "version 1.0", and since research was stopped

[1] Around 1990 that meant a 386 machine with 4 – 8 MBytes of RAM and a 40 Mbyte hard disk.

very much overnight, no effort was made to wind things up. This means that many things were only half-implemented or ad hoc, and while the English parser certainly worked quite well, I make no particular claims to elegance or linguistic correctness.[2]

2 The Role Of Punctuation In The System

Punctuation was recognized as an important part of natural written text from the very beginning of the project. Already in the 1987 prototype there were some rules for commas and periods. In fact, I do not think the option of not treating punctuation at all was ever considered. However, while punctuation certainly is important, it is only one of many interesting areas handled by the system (other foci include coordination, unbounded dependencies and numeral expressions).

Briscoe (1994) claims that "there are three major problems to be overcome in the development of a practical automatic parsing technology: text chunking, ambiguity resolution, and undergeneration." With limited computer resources, I would add a fourth: overgeneration. A strategy of stripping off the punctuation (and possibly using it as a filter afterwards) simply would not work, since the objects created by the comparatively deep analysis in this system are fairly large, and for a sentence of any reasonable length the computer memory would quickly fill up with useless partial parses long before any complete parse is ever reached. The processing (i.e. the rules) must be tightly constrained, and useless partial parses recognized as such and pruned as early as possible. This kind of constraining seems to be one of the main functions of punctuation.[3] Consequently, there is no separate punctuation level; instead, punctuation is integrated with the grammar at all levels, from morphology (e.g. hyphens and apostrophes) to the sentence (e.g. periods).[4] As a result, this system typically finds a single parse (or at most a handful) even for sentences with 20 – 30 words. The drawback with tight constraints, however, is that the system becomes extremely sensitive to even slightly ill-formed input, and a single misplaced comma may have catastrophic results. But this kind of oversensitivity is a general problem with the whole unificational approach, and is no worse for punctuation than for other lexical or phrasal categories. (In fact, most parsing errors are due to incorrectly flagged dictionary entries.)

3 Punctuation Categories

Since the system runs under Japanese operating systems, the term *punctuation characters* here refers to all the printable, non-alphanumeric characters in Japanese 7-bit ASCII code.[5] Just as there is no separate punctuation level, there is no general punctuation category. Rather, the punctuation characters are divided into a number of categories, on a par with (and in fact partially overlapping with) the standard lexical categories. Some of the characters belong to several categories, sometimes for technical or historical (rather than linguistic) reasons.

- **EOS** (End Of Sentence) **MARKS**: period, question mark, exclamation mark, ellipsis, and, perhaps controversially, colon and semicolon.

  All of these can also be RIGHT DELIMITERS, in which case they have a feature that lets them unify with a left comma or dash. (This probably means that the EOS MARK category itself could be eliminated altogether.)

- **SEPARATORS**: comma and (probably incorrectly) dash.

- **LEFT DELIMITERS**: left parenthesis, left square, angle and curly bracket, comma and dash.

- **RIGHT DELIMITERS**: right parenthesis, right square, angle and curly bracket, comma, dash, and all the characters listed as EOS MARKS above.

[2] I do not want to create the impression that the Japanese side of the system does not work very well, but since the system was never finished the English and Japanese components are often somewhat out of phase. This means that the system as a whole in its current state performs significantly less well than the English parser on its own.

[3] There are of course many other kinds of constraints as well, in particular a large set of lexical constraints that are included in the dictionary entries, percolated through the rules, and then executed when appropriate to let exceptions to general rules, or words with limited combinatorial properties, take care of themselves.

[4] Jones (1994) also advocates an integrated grammar, but his approach is rather different.

[5] Japanese 7-bit ASCII code differs from the U.S. version only in that it contains the yen symbol ¥ (which incidentally is not normally used in Japanese) instead of backslash. Japanese 8-bit ASCII, on the other hand, contains mostly *kana* characters in the upper half, and cannot be used for the present purpose.

The symmetrical characters have features that let them unify only with their corresponding half. Usually both delimiters have to be found in the same context-free rule. Commas and dashes can unify with each other or with any of the EOS MARK characters (currently, a left comma or dash cannot unify with a right bracket which is absorbing the otherwise required right comma/dash, but that is fixable).

- **LEFT and RIGHT QUOTES**: single and double

  The single quotation mark also doubles (no pun intended) as APOSTROPHE, which is only used to mark the genitive case for words ending in "s". Word-internal apostrophes (don't, can't, etc. as well as 's) are included in the lexical entries and are not treated separately.

- **OTHERS** (many of which are their own categories, further discussed below): hyphen, percent, slash, currency symbols, number sign, ampersand and ellipsis.

  For no apparent reason we seem to have left out plus and the equal sign (I suppose they never occurred in the test material), but they are easily added. The same goes for the remaining characters (∗, @, ˆ etc.) to the extent that they have any systematic function in the texts that are to be translated.

3.1 Colons and Semicolons

At first it might appear controversial to treat colons and semicolons as EOS marks (that is, to chunk the input in text-clauses rather than text-sentences[6]), but there are several good reasons for this.

First of all, note that in a text-sentence like

(1) Clause1; clause2.

clause1 does not constrain clause2 in any *syntactic* way.[7] There may certainly be semantic constraints, but those would usually require extremely sophisticated semantic processing far beyond the limits of this system and can thus be ignored for the present purpose.

Second, sentences including colons and semicolons are typically rather long and put a heavy strain on a computationally limited system. The clauses may parse perfectly when treated separately, but fail due to lack of memory when treated together.

Third, if the clauses are treated separately, an error in one of the clauses does not affect the parsing of the other clause, whereas if the they are treated together, the whole analysis would fail.

Also note that for colons *clause2* does not even have to consist of text, but might equally well be some kind of formula or an illustration!

The only obvious case where this treatment fails is when the colon or semicolon appears within parentheses:

(2) Clause1 (clause2; clause3) clause1.

will be treated (and fail) as

(2a) Clause1 (clause2;

(2b) clause3) clause4.

There may be other, highly marginal cases involving "promoted" list commas where text-clause chunking might fail, but there is no need to worry about those until an actual example is encountered, which seems unlikely.

Colons are also used at a lower level in time expressions like *11:30 p.m.* Currently, the system does not split up sequences of digits and punctuation characters, so *11:30* is handled as a single item. I am not aware of any lower-level use for semicolons.

[6] While I have borrowed some of the terminology in this paper from Nunberg (1990), it must be stressed that the research presented here was largely done before the publication of his important book, and so — for better or worse — was not influenced by it.

[7] Briscoe (1994) makes a similar observation, but does not go on to make the chunking break at the text-clause level.

3.2 Periods

There is a non-trivial chunking problem of distinguishing abbreviation periods from EOS periods. The naive approach is that a sentence break is signaled by a period followed by a space followed by a capital, but this will not work because of examples like *Mr. Smith*. The chunker is essentially pre-linguistic and has no access to dictionary information; hence, it has no way to tell that *Mr.* is an abbreviation that is likely to be followed by a proper noun. In the Bravice system, the problem is "solved" by requiring double spaces between sentences. Apparently, there is such a convention: Nunberg (1990) seems to take it for granted, and Quirk et al. (1985) say that "between the period and the capital [at the beginning of the next sentence], there will usually be a somewhat wider space than we normally have between words." I doubt, however, that the average user is aware of this (publishing) convention, and some pre-processing is usually necessary.

Decimal points are recognized as such, but not further analyzed. Sequences of digits and punctuation characters are treated as a kind of numbers. Parsing numbers might seem like overkill (and doing it in LFG certainly would be), but in fact many languages use a decimal comma and separate groups of three digits with points — the opposite of the English practice — and in Japanese the whole system of counting is different for large numbers, which are based on multiples of 10,000 (*man*) rather than 1,000, so that an expression like *6.2 million* becomes *620 man* in Japanese. For this reason, analysis of decimal expressions was on the agenda, but was never actually implemented.

3.3 Commas

Separator commas mostly appear in the coordination rules, of which there are many since coordination was another focus of research and since the metalanguage does not allow variables over categories. The comma between the penultimate conjunct and the conjunction is always optional, as are commas between attributive adjectives. Separator commas are also used in formulas for dates and the like.

Delimiter commas, like brackets, come in pairs, although for commas the left comma may be "implicit" sentence-initially, and the right comma may be absorbed by (the right delimiter homograph of) an EOS MARK sentence-finally. The system currently allows comma-delimited appositions, non-restrictive relative clauses, prepositional phrases and subordinate clauses in the usual places, as well as adverbial interpolations like *however*, which can occur in very many places in a sentence, but which are all taken care of (I think). Since commas have many functions, they may sometimes be needed at a lower level in the structure, in which case a feature is propagated along the right-most daughter to substitute for the right comma required at the higher level. Thus, a single right comma may correspond to several left commas.

In long lists of NPs, it can be very difficult to distinguish between separator commas and delimiter commas (that is, to determine whether an NP in the middle of the sequence is an element in the list, or a comma-delimited apposition of the previous element). For example, how many people are included in the following list?

(3a) Hashimoto Ryutaro, the Prime Minister of Japan, and Bill Clinton.

Since it seems generally impossible to define what kind of NPs can be appositions of what other kind of NPs, the parser has no way to tell. To avoid a combinatorial explosion for longer lists, the current rule allows an NP in the list to have a comma-delimited apposition only if the conjunct to the right of it also has an apposition (in line with the general tendency of moving "heavy" constituents to the right). This leads to a single parse for (3a) — which (at the moment of writing) unfortunately happens to be the wrong one. The problem disappears if the separator commas are promoted to semicolons, as in (3b), but that practice is hardly universal nowadays.

(3b) Hashimoto Ryutaro, the Prime Minister of Japan; and Bill Clinton.

There is also the rather common problem of "impressionistic" or prosodic commas. It would be possible to allow, say, an optional comma after a "heavy" subject NP, even though such a comma normatively speaking is incorrect. To the extent that errors are predictable, they can usually be handled in some fashion, but care must be taken not to let such rules wreak havoc with the parsing of ordinary correct sentences.

3.4 Parentheses

It may be possible (if difficult) to catalogue and treat the kinds of structures that can be delimited by commas. With parentheses, however, the situation is more unclear. Currently, the system allows the same sorts of parenthetical adjuncts that could also have been delimited with commas, with some differences in the constraints. For example, parenthetical phrases cannot appear sentence-initially, since the sentence rules look for a topicalized phrase followed by a right delimiter (that is, a comma or a dash for which the left delimiter can be implicit), and a left parenthesis is always explicit. (I have no particular objection to sentence-initial parentheses at the lexical level, which Nunberg (1990) rules out, but which the system would allow.) But in general the structures that can occur in parentheses are much freer than with commas. I am not sure whether that makes them easier or more difficult to handle. Adding rules for adjuncts that are OK in parentheses but would be ungrammatical or extremely unlikely with delimiter commas (for example, whole sentences or a single question mark) is not particularly difficult.

However, problems arise with parentheticals that are not adjuncts (or simple alternatives — such as this very parenthesis — which can be treated as a kind of coordination). Such sentences crucially have two distinct syntactic structures without being ambiguous in the conventional sense: one (what I would like to call the foreground parse) where the parentheticals are removed altogether, and one (the background parse) with the parentheticals inserted, possibly replacing some of the foreground material. For instance, in (4) (from Nunberg (1990), example 6.30) the NP structures are very different between the foreground and background parses.

(4) Clearly (any utterance of) the sentence *It is raining* will be true only if it is raining when the (utterance of the) sentence is spoken.

I have no idea how to treat such sentences, since the whole parsing methodology is geared towards finding one single parse, not two. Even if the two parses were found, for a machine translation system the problem would remain of how to compress the foreground and background parses into a corresponding target sentence.[8]

An interesting special case is parentheticals at the morphemic level:

(5) Insert the plug(s) into the corresponding socket(s).

Here, English syntax forces the nouns to be marked for number, but it is easy to conceive of a single functional structure for both the foreground and the background parses, in which the number features are unspecified. Sentences like this seem treatable, but further research is necessary.

3.5 Square, Curly and Angle Brackets

These are treated similarly to ordinary parentheses, although it is doubtful that curly and angle brackets are ever used like that. While square brackets may be used as higher-level parentheses, their most common use of seems to be for editorial fill-ins in quotes. Such fill-ins are not treated, though, and pose problems similar to non-adjunct parentheticals.

There is also the little-studied (?) area of bracketed numerals and quasi-numerals commonly used in academic texts for references and examples, such as (19), <2.27a>, [e], etc. General rules for these are not currently implemented, but it would be nice to do so.

3.6 Dashes and Hyphens

Dashes are rightly or wrongly treated exactly like commas, though dashes can probably only be delimiters. Hyphens are allowed for line breaks and in many word formation rules, particularly involving numeral expressions. The problem is distinguishing dashes from hyphens with ASCII input. There is a typing convention, adopted in this system, of typing (em) dashes as two hyphens, but once again many users will be unaware of this. It is probably better to distinguish them (and minus signs) according to the presence or absence of surrounding spaces, but there are borderline cases, such as *1987–96* or *1987 – 96*, which can be written both with and without spaces. The system currently looks for a hyphen in this context, but apparently it really should be an en dash. The obvious solution is to allow both hyphens and dashes in such cases.

In many other languages a dash is used to introduce direct speech, but this option seems unused in modern English (I wonder why), where quotation marks are the norm.

[8]That problem would be analogous to the translation of willful ambiguities or puns.

3.7 Quotes

Quotes were only half implemented when the research was stopped, and only quoted sentences and quoted NPs are allowed. Quote transposition works in some cases but not in others.

While it may or may not be possible to tell which structures can be delimited by parentheses or commas, it definitely is not possible for quotes. Almost anything goes: misspelt words, bad syntax, foreign languages, onomatopoeic expressions, etc. The quotation marks give a free license, and relieves the author of grammatical responsibility. It is doubtful whether any current linguistic theory provides a framework for parsing such sentences. Even if the quoted material is perfectly grammatical, it does not have to be a syntactic constituent at any level. This means that any attempt (including, of course, the system described here) to treat quotes with context-free rules as part of the grammar is doomed to fail in a large number of cases. It might be a better idea to mark all the words within quotes in some fashion, and then parse the sentence as if the quotation marks were not there. Note that for machine translation it is not sufficient to just attach the quotation marks to the first and last words of the quote, since the word order might be completely different in the translation.

3.8 Currency Symbols

Currency symbols are a very clear example of where written language differs from spoken language. Although the symbols correspond to words, their syntax is completely different.

3.9 Slash, Ampersand and Ellipsis

These are treated as conjunctions: "/" like *or*, "&" like *and*, and "..." like *etc.*

3.10 Number Sign and Percent

"#" is treated like the noun *number* (in certain contexts) and "%" like *per cent.*

4 Japanese Punctuation — A Digression

The Japanese punctuation system is different from the English in many interesting ways, and would be worth a deeper study of its own. I am not aware of any monograph on the subject, but at least Takebe (1979), a thick reference volume on the Japanese writing system, devotes some eight pages to it. Much of the following exposition derives directly from his work.

Japanese is written with a combination of four different types of characters,[9] with no spaces between words or phrases. But while the characters used for the lexical material remain unchanged, the sets of punctuation characters are different depending on whether the text is written vertically or horizontally![10]

Historically speaking, the Japanese punctuation system is a fairly recent invention. It was originally introduced as a way of annotating older texts written entirely in Chinese characters to make those texts easier to understand and to read aloud. That is, the punctuation was done by the reader, rather than the author. When the modern system of using both *kanji* and *kana* characters took over, the old practice of not using any punctuation remained unchanged. In fact, even in the late 19th century the prevailing attitude among authors seems to have been that "if it is necessary to insert punctuation characters to be able to understand the text, then the style is no good to begin with." It was only at the beginning of the 20th century that punctuation started to become widely used in official documents. By now, of course, punctuation is omnipresent and just about as standardized as for any other modern printed language.

Japanese has a number of punctuation characters that closely correspond to the English ones, such as 。 (*maru*: EOS period; an "English" point is used as decimal point and sometimes as EOS period in certain kinds of horizontal writing), 、 (*ten*: separator and delimiter comma, although comma insertion in Japanese seems to be based more on prosodic than on syntactic principles — not surprising, given the historical perspective; in horizontal writing *ten* is often replaced by an "English" comma), 「 」 (*kagi*: quotation marks), parentheses, long and short dashes, and a six-dot ellipsis sign. Colons are only used in horizontal writing.

[9] Almost any textbook on the subject would mention only three systems — *kanji*, *hiragana* and *katakana* — but in fact any Japanese newspaper or magazine also contains a very large number of "Roman" letters, very much integrated with the rest of the characters.

[10] The vertical mode is the default for newspaper text, fiction, etc., while the horizontal mode is used for most scientific and technical writing and as a layout option in magazines, etc. Often both modes are used on the same page. Most of the punctuation characters are rotated by 90 degrees between the two modes, probably for graphical reasons. Also, some characters only occur in the horizontal mode.

Semicolons, however, are not used at all. Since Japanese has a question particle, question marks are strictly speaking not necessary, but they occur frequently anyway to mark an interrogative intonation, especially in renditions of dialogue. Similarly, there are several particles that can be interpreted as exclamation particles.

There are also two specifically Japanese punctuation characters: ∼ (*namigata*), a kind of dash meaning *from ... to ...*, and the idiosyncratic ・ (*nakaten*), a multi-purpose separator which sometimes corresponds to a slash in English with a meaning like *and/or*, sometimes just to a space, such as between the first and last names in the personal names of foreigners.

5 Translating Punctuation

In the Bravice system, the English punctuation characters are usually transferred to their Japanese equivalents: an EOS period is transferred to *maru*, a comma to *ten*, etc. I am not sure this is a good idea, though. Probably punctuation characters only ought to be translated to the extent that they have some kind of content, and all punctuation characters are not created equal in this respect. Some of them closely correspond to spelled-out lexical items and have the same meaning, while others seem to have no particular meaning at all. But since the punctuation characters belong to a number of different categories, there is no reason why they all should have to be treated in the same way. Some of them, notably commas and EOS periods, seem to have an exclusively syntactic function, and once the English sentence is parsed they have no further use and should (probably) be discarded. The Japanese target sentence could then be generated with its own punctuation according to Japanese syntactic and stylistic conventions, quite independently of the punctuation in the original English sentence.

Unfortunately, things are more complicated than that. There is a sense in which the choice to use a comma, say, rather than not using any comma, is meaningful, and the relative frequency of punctuation characters can be consciously exploited as a stylistic device. For a human translator, particularly of fiction, this can be a fairly important aspect of the text, and one might be forced to insert (or delete) commas in different places than in the source text just to maintain the relative frequency when the syntactic structures do not match very closely between the two languages. For a machine translation system, though, that would be asking too much, but this is an interesting topic that requires further study.

6 Conclusion

As I hope this overview has made clear, it is very much possible to treat punctuation just like any other lexical and phrasal categories in a unificational framework. The problems and limitations of this approach to punctuation are basically the same as for the linguistic framework itself, in particular the difficulty of handling ill-formed input.

Acknowledgement

Thanks to MT Laboratories, Inc., Tokyo, the current owners of the system, who helped me test the system again after a 5-year hiatus, and who are searching for sponsors to continue this project.

References

Briscoe, Edward. 1994. Parsing (with) Punctuation etc. Rank Xerox Research Laboratory MLTT-TR-002.

Jones, Bernard. 1994. Exploring the Role of Punctuation in Parsing Natural Text. In *Proceedings of COLING 94*. Kyoto, Japan.

Nunberg, Geoffrey. 1990. *The Linguistics of Punctuation.* CSLI Lecture Notes 18, Stanford, CA.

Quirk, Randolph; Greenbaum, Sidney; Leech, Geoffrey; and Svartvik, Jan. 1985. *A comprehensive grammar of the English language.* London, UK: Longman.

Takebe, Yoshiaki. 1979. *Nihongo no hyouki.* Tokyo, Japan: Kadokawa shoten.