A Non-deterministic Tokeniser for Finite-State Parsing

Jean-Pierre Chanod
Rank Xerox Research Centre
Grenoble Laboratory
6 Chemin de Maupertuis
F-38240 Meylan
France
Jean.Pierre.Chanod@xerox.fr

Pasi Tapanainen
University of Helsinki
Research Unit For Multilingual Language Technology
Department of General Linguistics
P.O. Box 4
Finland
Pasi.Tapanain@ling.Helsinki.fi

Abstract. This paper describes a non-deterministic tokeniser implemented and used for the development of a French finite-state grammar. The tokeniser includes a finite-state automaton for simple tokens and a lexical transducer that encodes a wide variety of multiword expressions, associated with multiple lexical descriptions when required.

1 Introduction

Usually tokenisation has been seen as an independent process [5, 9] in natural language processing. In many parsing systems the tokenisation has had little attention and, especially when parsing English, tokens are often supposed to be sequences of letters between two blanks.

In our approach the tokenisation is a firm part of the morphological analysis. Our tokens are defined for the needs of a syntactic parser (i.e. they are the basic components of the parsing). This leads us to a large collection of different tokens, like:

- a simple word,
- several words forming one token as in *a priori*,
- a same string ambiguously producing one or several tokens as in *de même*,
- a sequence of words that may have significant variation, like *il va y avoir bientôt cinq ans*, and
- a simple word that may become two tokens, e.g. *du*.

Our system has the following properties. The tokenisation is able to use lexical information, i.e. it recognises the tokens that are recognised by the morphological analyser. Also, tokens may be ambiguous, i.e. a sequence of characters may contain ambiguously one or more tokens. And what is important because we are using it in a development environment, the system is reasonably fast to update.

We describe a tokenisation process that uses two distinct finite-state transducers. Tokenisation is not a totally independent process, but closely related to the morphological analysis. The tokens are not described in detail here. What kind of tokens are needed depends on the finite-state network based syntactic analyser for French which has been developed during the last few years [3, 4].[1] For related work, see [1, 8, 10, 12].

Proceedings of the ECAI 96 Workshop
Extended Finite State Models of Language
Edited by A. Kornai.

2 Non-deterministic tokenisation

As the first step in the analysis, a tokeniser segments the input sentence into tokens. In many applications, it is assumed that at this level of processing there is no ambiguity. Karttunen [6] describes the compilation of unambiguous tokenisers from direct replacement expressions. Such tokenisers are now implemented for various languages. They consist of a single finite-state network that produces unique output.

While this approach is acceptable for certain applications such as part of speech disambiguation, it may not be satisfactory for more refined processing, such as syntactic analysis at a sentence level.

For instance, in French, the string *de même* (similarly) can be treated as a single token (an adverb), or as a sequence of two independent tokens: the preposition *de* (of) followed by the adjective *même* (same) as in *de même format* (of (the) same format). If a wrong reading is arbitrarily selected during the tokenisation, it may lead to inconsistent and incorrect syntactic analyses.

3 Two networks for tokenisation

In order to produce non-deterministic outputs during the tokenisation phase, we propose a novel architecture: it uses two finite-state networks rather than just one: a simple automaton (the basic tokeniser) and a dedicated lexical transducer that describes multiword (MW) expressions.

The *basic tokeniser* identifies general sequences of characters (combinations of any number of letters, digits and to some extent punctuation signs), without considering whether they belong to the language or not. Basically, it recognises any string that does not have blanks.

The *multiword tokeniser* is actually a MW lexicon, as opposed to the basic tokeniser that is a simple finite state network. The multiword tokeniser recognises the long tokens not accepted by the basic tokeniser. Such long tokens consists of,

[1] See: http://www.xerox.fr/grenoble/mltt/reports/home.html.

e.g., a list of known (possibly ambiguous) multiwords and regular expressions for punctuation and numeric expressions, such as *234 000*, and also for adverbials of time like *le 1er mai*. The latter simplifies the syntactic analysis; i.e. we don't want to write rules for constructions that we can list.

The tokenising networks are used in the following way:

1. We read characters from the input stream; whenever the multiword tokeniser accepts a string of characters, we use this tokeniser. We extract the longest match and obtain the morphological analysis at the same time.
2. Otherwise, we extract the longest match that the basic tokeniser accepts and analyse the output with other lexicons (standard lexicon, guesser, etc.).
3. The found token is removed from the input stream.

3.1 An example

Let us look at an example sentence[2] extracted from the *Le Monde* corpus.

> De même, "Port-Mariane sera ce que le marché en fera", dit Raymond Dugrand.

The tokenisers divide the sentence into fourteen parts:

> De même, " / Port-Mariane / sera / ce / que / le / marché / en / fera / ", / dit / Raymond / Dugrand / .

The MW lexicon recognises the sequences of punctuation (`",` above), and it also knows how such sequences should be analysed. An interesting token is the first one that is also recognised by the MW tokeniser. Besides the word sequence also the punctuation (and the blank characters) belong to the recognised sequence.

3.2 Punctuation and multiwords

The punctuation has its own mini-lexicon that is included in the MW lexicon. Basically, the punctuation lexicon contains all the punctuation characters repeated any number of times (and multicharacters like three dots ...) and transformed into themselves. Besides, blank characters are also included, and they are transformed into null. Finally, various word boundaries are inserted to the output. A simplified regular expression for this could be:

```
[.:. | !:! | % :0]+ [0:./ | 0://]
```

where `.:.` denotes that a dot is transformed into itself, `% :0` that a blank is removed, and `0://` that an (ambiguous) sentence boundary `//` is added to the output. This provides a mechanism for tokenising and producing the same word boundaries (or token boundaries) for *a,b* and *a, b*.

The punctuation itself is mostly just a curiosity but when we are dealing with the multiword expression, the punctuation lexicon becomes important. Consider, e.g., the adverbial of time *le 1er mai* (the first of May). Let us suppose that we want to recognise this as one token. Then we cannot just add it to the MW lexicon. The reason is that the tokeniser uses the longest match it founds from the MW lexicon and we could misanalyse a sequence *le 1er maire* (the first mayor). There are two ways to fix this: (1) we may add also the *le 1er maire* to the MW lexicon, or (2) we simply concatenate the punctuation lexicon after the multiwords in the lexicon. This way we obtain the result in the example above. In this case, the tokenisation is actually resolved after the morphological analysis.

4 Combining the tokenisers

In theory, the basic tokeniser and the MW lexicon could be composed into a single transducer. Let `T` denote the basic tokeniser, `M` the multiword lexicon, `A` the morphological analyser and `G` a guesser that gives every string an analysis. The tokenisation and the morphological analysis could then be expressed as one transducer, namely $\texttt{M} \cup [\ \texttt{T} \circ \texttt{A}\ ] \cup [\ \texttt{T} \circ [\ \texttt{G-A}\ ]]$ where $^\circ$ denotes the composition and `G-A` the transducer that does not accept the (input) strings accepted by `A`.

However, such a compilation easily leads to time and space problems, while the two independent networks remain reasonable compact even when the number of encoded MW expressions is high (typically, several thousands, not considering cyclic expressions such as dates).

5 Ambiguous tokens

For instance, in the case of *de même*, one gets the following two readings at the lexical level:

(1) de Prep ./ même InvGen SG Adj
(2) de_même Adv MW

The first reading includes the symbol `./` which represents a word boundary between the preposition *de* and the singular adjective *même*, while the second reading is a single token, marked with a MW tag.

Ambiguous MWs (with respect to tokenisation) cover various combinations such as preposition + adjective in the *de même* example, or adjective + noun as in *bon marché* (cheap) as a MW adverbial, but also *good market* when analysed as two independent tokens:

(1) bon Masc SG Adj ./ marché Masc SG Noun
(2) bon_marché InvGen InvPL Adj MW
(3) bon_marché Adv MW

More complex situations may occur when the decomposition of the MW into more than one token leads to many combinations. This is for instance the case with *bien que*, a MW connective than means *although*. When *bien que* is split into two tokens, *bien* can be a noun, an adjective or an adverb, and *que* a conjunction or an accusative relative pronoun. Besides, the type of word boundary between *bien* and *que* may vary, depending on the type of clauses (simple clause, embedded clause, etc.) to be found between the two tokens. In the end, one gets about 200 lexical analyses for *bien que*, in addition to the MW connective reading.

6 Components of the MW lexicon

The MW lexicon results from the union of various lexical resources:

[2] In English: Similarly, "Port-Mariane will be what the market will do out of it", said Raymond Dugrand.

- MWs from the basic lexicon
One component of the MW lexicon results from the extraction of MWs encoded in our basic French morphological analyser [7, 2]. This represents about 8000 MWs (6200 nouns, 1000 adverbs, 350 grammar words, 200 adjectives).
- Alternate readings
Another component of the MW lexicon originates from a transducer that encodes alternate readings for MWs, i.e. readings where the string is not recognised as a single token, but rather as a sequence of independent tokens, as mentioned above for *bien que* or *de même*.
Such alternate readings are described using expressions [11, 13] defined for that purpose. These regular expressions extract words from the basic morphological analyser, and combine them as required. The complete lexical reading results from the concatenation of the contiguous extracted words, while the relevant word boundaries are inserted in-between. For instance, one alternative reading of the MW *bien que* results from the extraction of the adverb *bien* and the connective *que*, both being concatenated with any word boundary in-between (as any clause may start before *que*).
- Miscellaneous readings
Another component of the MW lexicon originates from a lexical transducer which encodes MWs that are not found in the basic morphological analyser. This covers a wide range of phenomena, such as misspelt expressions, e.g. *à priori*, domain specific terminology, names, idioms, or expressions that constitute a challenge for the parser by going against the predictions of the general syntax, e.g. *fin mai* (lit. end May), *le pourquoi et le comment* (the why and the how).
- General regular expressions
Another way of adding information to the MW lexicon is to use general regular expressions that combine specific words, e.g. nouns appearing in time adverbials, and more generic words extracted from the lexicon, e.g. prepositions, numbers, etc.
This is how we encode time MWs like dates, e.g. *lundi 21 janvier 1794*, or verbal phrases that behave as adverbials, e.g. *il va y avoir bientôt cinq ans* (lit. there will be soon five years), where some inserted elements belong to predefined sublexicons (e.g. numerals, adverbials). For example:

  il `[` *va y avoir* | *y a* | *y aura* `] Adverb Numeral [` *jours* | *mois* | *années* `]`.

7 Conclusion

We described a non-deterministic tokeniser integrated in a finite-state parser. The tokeniser being non-deterministic means there is no loss of information when a given string can be seen either as one token or decomposed into several tokens. The tokeniser allows us to encode in a compact fashion a wide range of multiwords, ranging from dates, names, terminologies, complex adverbials, idioms and more general phrases that are difficult to parse. This in turn, improves the accuracy of parsing.

REFERENCES

[1] Steven P. Abney, 'Parsing by chunks', in *Principled-Based Parsing*, eds., R. Berwick, S. Abney, and C. Tenny, Kluwer Academic Publishers, Dordrecht, (1991).

[2] Jean-Pierre Chanod, 'Finite-state composition of french verb morphology', Technical Report MLTT-005, Rank Xerox Research Centre, Grenoble Laboratory, France, (1994).

[3] Jean-Pierre Chanod and Pasi Tapanainen, 'A lexical interface for finite-state syntax', Technical Report MLTT-025, Rank Xerox Research Centre, Grenoble Laboratory, France, (1996).

[4] Jean-Pierre Chanod and Pasi Tapanainen, 'Rules and constraints in a finite-state grammar', Technical Report MLTT-024, Rank Xerox Research Centre, Grenoble Laboratory, France, (1996).

[5] Gregory Grefenstette and Pasi Tapanainen, 'What is a word, what is a sentence? problems of tokenization', in *The 3rd International Conference on Computational Lexicography*, pp. 79–87, Budapest, (1994).

[6] Lauri Karttunen, 'Directed replacement', in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, USA, (1996).

[7] Lauri Karttunen, Ron Kaplan, and Annie Zaenen, 'Two-level morphology with composition', In *Proceedings of the Fourteenth International Conference on Computational Linguistics COLING-92*, volume I, pp. 141–148, Nantes, France, (1992).

[8] Kimmo Koskenniemi, Pasi Tapanainen, and Atro Voutilainen, 'Compiling and using finite-state syntactic rules', In *Proceedings of the Fourteenth International Conference on Computational Linguistics COLING-92*, volume I, pp. 156–162, Nantes, France, (1992).

[9] David D. Palmer and Marti A. Hearst, 'Adaptive sentence boundary disambiguation', in *Proceedings of the 4th Conference on Applied Natural Language Processing*, pp. 78–83, Stuttgart, Germany, (1994).

[10] Emmanuel Roche, *Analyse syntaxique transformationnelle du français par transducteurs et lexique-grammaire*, Ph.D. dissertation, Université de Paris 7, 1993.

[11] Frédérique Segond and Pasi Tapanainen, 'Using a finite-state based formalism to identify and generate multiword expressions', Technical Report MLTT-019, Rank Xerox Research Centre, Grenoble Laboratory, France, (1995).

[12] Max Silberztein, *Dictionnaires électroniques et analyse automatique de textes. Le système INTEX*, Masson, Paris, 1993.

[13] Pasi Tapanainen, 'RXRC finite-state compiler', Technical Report MLTT-020, Rank Xerox Research Centre, Grenoble Laboratory, France, (1995).