JON MILLS
University of Luton
Luton, UK
E-Mail: jon.mills@luton.ac.uk

Screffva: A Computational Architecture for Bilingual Corpus Lexicography

Abstract

This paper describes the implementation of *Screffva,* a computer system written in Prolog that employs a parallel corpus for the automatic generation of bilingual dictionary entries. *Screffva* provides a lemmatised interface between a parallel corpus and its bilingual dictionary. The system has been trialled with a parallel corpus of Cornish–English bitext. *Screffva* is able to retrieve any given segment of text, and uniquely identifies lexemes and the equivalences that exist between the lexical items in a bitext. Furthermore the system is able to cope with discontinuous multiword lexemes. The system is thus able to find glosses for individual lexical items or to produce longer lexical entries which include part–of–speech, glosses and example sentences from the corpus. The corpus is converted to a Prolog text database and lemmatised. Equivalents are then aligned. Finally Prolog predicates are defined for the retrieval of glosses, part–of–speech and example sentences to illustrate usage. Lexemes, including discontinuous multiword lexemes, are uniquely identified by the system and indexed to their respective segments of the corpus. Insofar as the system is able to identify specific translation equivalents in the bitext, the system provides a much more powerful research tool than existing concordancers such as *ParaConc*, *WordSmith*, *XCorpus* and *Multiconcord*. The system is able to automatically generate a bilingual dictionary which can be exported and used as the basis for a paper dictionary. Alternatively the system can be used directly as an electronic bilingual dictionary.

Introduction

There are a number of systems available which are capable of producing concordances from parallel corpora. Examples of such systems include *ParaConc* (see Barlow 1995), *Wordsmith Tools* (see Berber Sardinha 1996), *XCorpus* (see Bonhomme & Romary 1997) and *Multiconcord* (see St. John & Chattle 1998). With such systems, lemmatisation is usually achieved by entering all the variant forms for the lexeme under investigation. However this method achieves only partial lemmatisation as it is also necessary to disambiguate homographs. Wild cards are often employed to implement a fuzzy search for a particular lexeme; for example **tak*** will find **take**, **takes**, **taking** and **taken**, but not **took**. Use of wildcards also frequently finds words which are not part of the paradigm under investigation. Lemmatisation methods such as these are not precise and post–editing of concordances is, therefore, usually necessary in order to remove unwanted items that are not part of the paradigm being investigated.

Text alignment is usually carried out at the rank of sentence, so that, for example, sentence nine in the first text is equivalent to sentence nine in the parallel text. However alignment at sentence level may prove to be problematic because a translator may translate one sentence by two or more sentences. Alternatively texts can be aligned at the rank of paragraph, so that paragraph nine in one language is equivalent to paragraph nine in its parallel text. Whether alignment takes place at the rank of sentence or paragraph, concordances do not specifically identify translation equivalents of the particular lexemes under investigation. Instead entire sentences which provide the context for the translation equivalents are given.

Existing tools for working with parallel corpora seem, then, to lack two things, firstly the means to uniquely identify lexemes, and secondly the means to identify those items which share translation equivalence in a bitext. These problems are interrelated since it is between lexemes (rather than the graphical words) of a language pair that equivalence tends to exist. In order to solve these problems, firstly the system must be able to find and retrieve any given segment of text; secondly the bitext must be fully lemmatised; and thirdly the bitext must be aligned at the rank of lexical item.

Screffva consists of a number of modules: the tokenisation module, VOLTA – the lemmatisation module, the text alignment module, the corpus, and the dictionary.

Critical Tokenisation

Not all segments are tokens. A token is a segment that corresponds to a particular linguistic unit on the scale of rank. Thus a text may be said to consist of morpheme–tokens, word–tokens, phrase–tokens, clause–tokens, etc.. For the purposes of lexicography, we are usually concerned with tokens at the ranks of morpheme, word and multiword lexeme. Such tokens correspond to lexical items, and I

shall call these **lexical tokens**. So the set, *A*, of all tokens in a text is a subset of the set, *B*, of all possible segments in that text :

$$A \subset B \ .$$

Mills (1998) describes how critical tokenisation may serve as a basis for the construction of text databases in the Prolog programming language. It is necessary to represent a text in such a way that both individual tokens and individual types can be identified. Segments and points form a basis for tokenisation. A critical point in a text is that which delimits two adjacent segments. Conversely a critical segment of text is that which is delimited by two points in the text. Segments are, thus, located in time and space. Segments can be observed, pointed to and given unique names. Segments and strings are not the same. Segments are instances of strings. Two segments may be instances of the same string. Segments may, thus, correspond to tokens, whereas strings correspond to types. Figure 1 shows how critical tokenisation can be applied to a stretch of text.

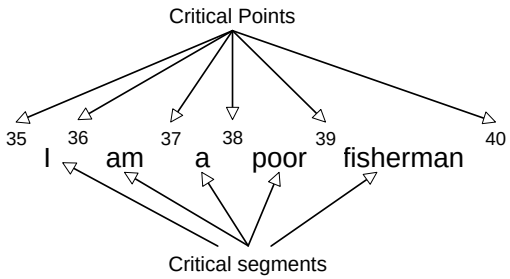

Figure 1

Figure 2 shows how the principles of critical tokenisation may be applied to create a Prolog database from this same stretch of text. The text is arranged vertically with each token on a separate line. Each token is represented by a three place predicate. The first and second arguments are the critical points which define the segment covered by the token. The third argument is the word type of which the token is an instance.

```
type((35, 36), 'I' ).
type((36, 37), am ).
type((37, 38), a ).
type((38, 39), poor ).
type((39, 40), fisherman ).
```

Figure 2

With the text in the form of a database it is possible to conduct various types of search. For example, if one wants to know what word type is found between critical points 36 and 37, one sets the goal,

```
?– type((36,37),T).
```

The system returns,

```
T = am
```

Conversely, if one wants to know in what text positions the word type *am* is found, one sets the goal,

```
?– type((X,Y),am).
```

The system returns,

```
X = 36
Y = 37
```

If one wants the phrase found between critical points 35 and 40, one sets the goal,

```
?– get_segment((35,40),Segment).
```

The system returns,

```
Segment = [I, am, a, poor, fisherman] .
```

This system thus provides the requirement of being able to find and retrieve any given segment of text.

Lemmatisation

Lexical items are selected from the corpus on which the dictionary is based whilst simultaneously the dictionary provides information concerning the lemmatisation of the corpus. The processes of dictionary lemmatisation and corpus lemmatisation are, therefore, interdependent. An ideal system for

corpus lexicography is one in which the corpus database and the dictionary are interactive. Mills (1999) shows how a logical approach to the lemmatisation of computational lexica may be implemented in Prolog. The counterpart of a Prolog dictionary is the lemmatised Prolog text database.

For the purposes of corpus lemmatisation, the lemma should ideally uniquely identify the lexical item. Part of speech tagging goes some way towards this. Tagging with the baseform further disambiguates the item. Clauses are added to the Prolog text database to represent lemmatisation. VOLTA (Mills 1995) is a program that lemmatises a corpus by relating word tokens as they are encountered in the corpus to a lexicon that has capacity to expand as new items are are added. Figure 3 shows the lemmatisation of the extract that we saw in Figure 2.

```
lemma( (35, 36), ('I', pron) ).
lemma( (36, 37), (be, v) ).
lemma( (37, 38), (a, art) ).
lemma( (38, 39), (poor, adj) ).
lemma( (39, 40), (fisherman, n) ).
```

Figure 3

Within the bitext, the English word FISHERMAN is translated by the Cornish multiword lexeme DEN AN PUSKES (literally DEN = *a man,* AN = *of the,* PUSKES = *fishes*). Figure 4 shows the tokenisation and lemmatisation of the Cornish translation of the English phrase given in Figure 2. The system is very flexible. Not only can the multi–word lexeme, DEN AN PUSKES, be lemmatised as a single lexical item but its component lexemes DEN, AN and PYSK may be simultaneously individually lemmatised. The appearance of the lexeme, PYSK, in its plural form, *puskes,* within the multiword lexeme, DEN AN PUSKES, does not present a problem to the system. Nor does it present a problem that, due to interruption by the lexeme, BOGHOSEK, the multiword lexeme, DEN AN PUSKES, is, in this instance, discontinuous.

```
type( (29, 30), thearra ).
type( (30, 31), vee ).
type( (31, 32), dean ).
type( (32, 33), bodjack ).
type( (33, 34), an ).
type( (34, 35), puscas ).
```

```
lemma( (29, 30), (bones, v) ).
lemma( (30, 31), (vy, pron) ).
lemma( (31, 32), (den, n) ).
lemma( (32, 33), (boghosek, adj) ).
lemma( (33, 34), (an, art) ).
lemma( (34, 35), (pysk, n) ).
lemma( (31, 35), ('den an puskes', n) ).
```

Figure 4

The lemma database thus records that the segment between critical points 31 and 35 contains five lexemes: DEN between points 31 and 32, BOGHOSEK between points 32 and 33, AN between points 33 and 34, PYSK between points 34 and 35, and the multiword lexeme DEN AN PUSKES between points 31 and 35.

Alignment

Equivalence exists between the lexical units of a bitext rather than its word types. For this reason alignment takes place between lexical tokens rather than graphic word tokens. Equivalents may be entered into the system as 2 place predicates in which the first argument specifies the critical points that bound the Cornish lexical unit, whilst the second argument specifies the critical points that bound the lexical unit which is its English translation.

```
equivalent( (29,30), (36,37) ).
equivalent( (30,31), (35,36) ).
equivalent( (31,35), (39,40) ).
```

Figure 5

Thus the Cornish lexeme, BONES, found at token (29, 30), is translated by the English lexeme, BE, found at token (36, 37). Similarly the Cornish lexeme, VY, found at token (30, 31), is translated by the English lexeme, I, found at token (35, 36). And the Cornish multiword lexeme, DEN AN PUSKES, found at token (31, 35), is translated by the English single lexeme, FISHERMAN, found at token (39, 40).

It is important to note that this alignment refers to the lexemes listed in the lemma database and does not refer to the types listed in the original tokenisation. If it did, then the English type, *fisherman,* found at token (39, 40), would be the translation of the Cornish phrase, *dean bodjack an puscas*, found at token (31, 35), which is not the case. *Dean bodjack an puscas* translates into English as *a poor fisherman*.

A number of predicates are defined which enable the corpus to be used like a dictionary. For example, the predicate, *gloss_lemma/2*, finds the English gloss for a Cornish item or vice versa. The predicate, *entry/1*, displays the full dictionary entry for the requested item (see

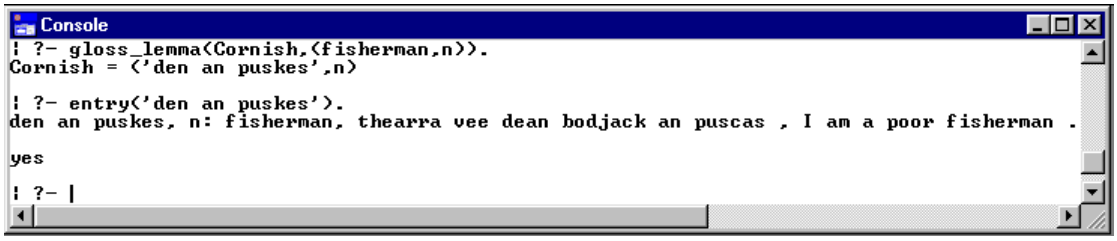

Figure 6).

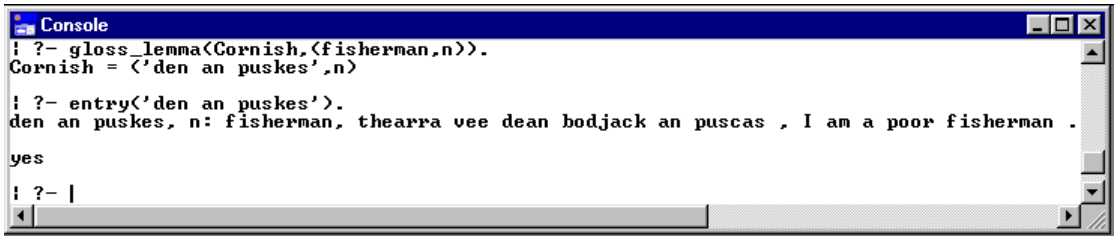

Figure 6

Discussion

The system is able to retrieve any given segment of text, and uniquely identifies lexemes and the equivalences that exist between the lexical items in a bitext. Furthermore the system is able to cope with discontinuous multiword lexemes. The system is thus able to find glosses for individual lexical items or to produce longer lexical entries which include part–of–speech, glosses and example sentences from the corpus. Insofar as the system is able to identify specific translation equivalents in the bitext, the system provides a much more powerful research tool than existing concordancers such as *ParaConc*, *WordSmith*, *XCorpus* and *Multiconcord*. The system is able to automatically generate a bilingual dictionary which can be exported and used as the basis for a paper dictionary. Alternatively the system can be used directly as an electronic bilingual dictionary.

References

Barlow, Michael (1995) "A Guide to ParaConc" Available http://www.ruf.rice.edu/~barlow/pc.html .

Berber Sardinha, A. P. (1996) "Review: WordSmith Tools" *Computers & Texts* No. 12 p. 19. ISSN 0963–1763.

Bonhomme, Patrice & Laurent Romary (1997) *XCORPUS – Version 0.2: A Corpus Toolkit Environment: User Manual* Available: http://www.loria.fr/projets/XCorpus/manual/ .

Mills, Jon (1995) "Computer Assisted Lemmatisation of a Diachronic Corpus of Cornish" *Proceedings of TCIPA II, University of Luton, 7 July 1995*. pp. 9.1–9.16.

Mills, Jon (1998) "Lexicon Based Critical Tokenisation: An Algorithm" *Actes EURALEX'98 Proceedings: Papers Submitted to the Eighth EURALEX International Congress on Lexicography in Liège, Belgium* Liège: University of Liège, English and Dutch Departments. pp. 213–220.

Mills, Jon (1999) "A Logical Approach to the Lemmatisation of Computational Lexica" Proceedings of the Sixth International Symposium on Social Communication, Santiago de Cuba, 25–28 January 1998 . pp. 598–605.

St. John, Elke & Marc Chattle (1998) "Review of Multiconcord: The Lingua Multilingual Parallel Concordancer for Windows" *ReCALL Newsletter* No 13, March 98 . ISSN 1353–1921.