A PHRASE-RETRIEVAL SYSTEM BASED ON RECURRENCE

Magnus Merkel, Bernt Nilsson & Lars Ahrenberg

Department of Computer & Information Science
Linköping University
S-581 83 Linköping
SWEDEN
email: {mme,bkn,lah}@ida.liu.se

Abstract

The paper describes a simple but useful phrase-retrieval system that primarily is intended as a support tool for computer-aided translation. Given no other input than a text (and a word list used for filtering purposes), the system retrieves recurrent sentences and phrases of the text and their positions. In addition the system provides information on internal and external recurrence rates.

1. Introduction

As any localiser knows, manuals and other types of documentation are often highly repetitive, i.e. many phrases, in fact even sentences and sometimes paragraphs, tend to recur in the text. This is a fact that can obviously be used to speed up translation. By combining two sets of data for a recurrent segment, viz. a record of the positions in the text where it is found, and a record of its translation (assuming it to be unique), all occurrences of the segment can be translated in one go. As the recurrence rate within one document (*internal recurrence*) can sometimes be higher than 25% (see Table 1), and even higher if the document is compared, say, with a family of documents or a previous version of the same product (*external recurrence*), the gains in speed, costs and consistency can be quite substantial (see Table 2 and 3). In fact, it seems that translation aids of this kind are already in use by some translation companies, see e.g. Language Industry Monitor (1994). Moreover, information on recurrent segments and their frequencies is useful in combination with any type of memory-based translation system, as it tells you what would be useful to have in the system's database(s).

To make the scheme work, however, you need a system that finds the recurrent segments of a given document for you. Moreover, to reduce the time required for manual inspection and correction of the generated list of segments, the system should only generate segments that are appropriate translation units. It is such a system that this paper is about.

The system, nick-named FRASSE, actually has two primary uses. On the one hand it can be used as a data acquisition tool for a memory-based translation system, as described above.

It then actually derives only half of the needed data, as the translations have to be acquired in some other way. On the other hand it can be used diagnostically to generate a profile for a given document, calculating internal and external recurrence rates and frequencies for recurrent segments. This profile can be taken as a basis for determining whether a translation memory should be used at all, and if so, with what (initial) content in its databases.

Apart from its use in computer-aided translation FRASSE could also be a useful tool for a lexicographer or a literary student. Most systems with a functionality similar to FRASSE's, such as Choueka (1988) and Lessard & Hamm (1991) have apparently been developed for such purposes. The contributions of FRASSE are that the search process is exhaustive given a text as input and that phrase retrieval can be done on untagged texts. Furthermore, there is a measuring function which calculates how much of a given text is made up from recurrent segments and sentences. Large texts can be split up into smaller subtexts and analysed separately for reasons of complexity, but the retrieved segments from each subtext can be compared with other data to find differences or to combine subtext data into data for the whole text. FRASSE has been used on texts of 1 million words and can be run on both UNIX systems and PCs.

Internal Recurrence	**SS1**	**SS2**
Total no of words	254,350	248,089
Sentence Recurrence	25 % (2,290 sents)	15 % (1,822 sents)
Segment Recurrence	29 % (1,824 phrases)	31% (1,911 phrases)
Sentence & Phrase Recurrence	43 %	39 %

Table 1. Internal recurrence for User's Guides of two versions of the same spreadsheet program, SS1 and SS2 (from Merkel 1992). The phrase segments have been revised manually and segments which do not function as translation units have been removed.

External Recurrence in SS2 relative to SS1	
Shared sentences	3,677
External Sentence Recurrence	20 %
Shared Segments	620
External Segment Recurrence	15 %
External Sentence + Segment Recurrence	31 %

Table 2. External recurrence in SS2 relative to SS1

Combination Internal/External Recurrence	**SS2**
Internal + External Sentences	33 %
Internal Sentences + External Sentences + External Segments	42 %
Internal Sentences + External Sentences + Internal Segments	52 %

Table 3. Combination of internal and external recurrence in SS2 (relative to SS1)

The rest of this paper is organised as follows. Section 2 gives a short presentation of our goals for FRASSE within the framework of a project on computer-aided translation that we are currently engaged in. We also define some of the notions we will use in the sequel. Section 3 describes the functionality of the system and section 4 illustrates how it can be used. In section 5 we compare FRASSE with some other similar systems and in section 6, finally, we discuss our plans to improve it.

2. On the use of recurrent segments in translation

A basic assumption underlying our translation scheme is that a recurrent segment in the normal case is a translation unit. This assumption does not always hold, however. If recurrence is the only criterion, we are likely to identify segments that should really be treated differently, because they have a different analysis in different contexts. The segment "the file" should be treated differently if it is used in the context of (1) "the file manager", (2) "the file menu" or (3) "the file" when it is translated into Swedish. In context (1) it would be translated as "Fil(hanteraren)", in (2) as "Arkiv(-menyn)" and in (3) as "filen". Also, as will be shown below, we will find a good deal of linguistic nonsense that needs to be filtered out. It is thus important to design the system so that only relevant recurrent segments are found, which means that we have to find some criteria that distinguish the relevant segments from the irrelevant ones. Related to this question is the level of abstractness of the segments; are they to be represented as unanalysed word segments, lemmatised word segments or perhaps something more general such as phrase patterns or phrase structure rules?

Another important question is whether consistency in translation of recurrent segments is always to be preferred, as it is for terminology, or whether there are situations when different translations should be used, even though the source segment is a recurrent unit of the source text. In a small pilot study on aligned parallel texts (Larsson & Merkel, 1994) we found a few cases where translators had chosen different translations for the same segment of the source text, but with no apparent harm.

English: *Select the port you want to use.*
Swedish 1: *Välj den port du vill använda.*
Swedish 2: *Markera den port du vill använda.*

Apart from the dimensions of consistency and variation, one may suspect that the use of fixed translations for recurrent segments, may affect the coherence of the target text. This is a question that we intend to investigate in our further work.

We end this section with some definitions of the terms that we use in the paper.

- A *segment* is a sequence of two or more consecutive words that does not cross sentence boundaries.
- A *recurrent segment* is a segment that occurs in at least two different sentences of a given text or corpus. For practical reasons, we often use a higher lower bound

than two for the number of occurrences, but there is no non-arbitrary way to fix this lower bound. Note that a recurrent segment as such may be a proper syntactic unit, or a collocation, but that it can be neither.

- A *phrase* is a segment constituting a syntactic unit with compositional meaning.
- A *phrase pattern* is a sequence of words and categories that a phrase can be analysed as (e.g. *Select <segment> from <segment>*).
- A *collocation* is a segment constituting a lexical unit of some sort.
- A *translation unit* is a simple or complex linguistic unit of a source language that can be associated with a set of corresponding units of a target language. Different instances of a translation unit in a text may have different translations, but it must be possible to assign it some corresponding instance in the target text.

3. The FRASSE system

The program is implemented in standard C and there are currently compiled versions for UNIX, Windows NT and OS/2. The following functionality is included in FRASSE:

3.1 Retrieval of recurrent sentences

Sentences are identified in a technical sense, i.e. they are identified by punctuation information and carriage return characters.

Input: Text

Output: A segment table with sentence types (as word strings), position (document identifier and a list of offsets) and number of occurrences for each sentence in the text.

3.2 Retrieval of recurrent segments

Recurrent segments are retrieved from the text after it has been split up into sentence types. Either you can input the text (Frasse will then create a table with sentence types, see 3.1) or one or more existing segment tables. The data structure for storing segments consists of a table of segments. Each entry in the table contains the following fields:

1. *The text segment*
2. *A list of locations* where each location contains a document identifier and a list of offsets inside each document.
3. *A parsed word string*, a sequence of references to the word table, see below. All text segments are first parsed to word strings so that comparisons of words can be done by just comparing pointers. (This field is not present in the output.)

In addition to the above fields, the data structure of the segment table holds a *word table*, i.e. a table of known words with frequencies.

Each recurrent segment is stored in only one entry, but has multiple locations with multiple offsets.

> **Input:** Text or a list of segment tables
> **Output:** A new segment table holding maximal segments from input.

All combinations of segments from the input table are compared pairwise. Each pair, <*s1*,*s2*>, is searched for common parts. A common segment is defined as all segments that contain the same consecutive words in both *s1* and *s2*. If the identified segment is long enough it is stored with all locations and adjusted offsets from both *s1* and *s2*.

We only consider segments that start at locations such that the immediately preceding words, if any, are different in *s1* and *s2*. A segment ends in the same manner. This means that only the longest possible segments are considered. Note that both *a b c d* and *a b c d e* can be regarded as maximal segments if they have different frequencies, i.e. if *a b c d* occurs 15 times and *a b c d e* 12 times they are both maximal segments.

The fact that only the longest possible segments for each pair are stored has the effect that there is a significant reduction of data, which makes the algorithm more efficient.

There is a filter that trims the segments at the head and tail of the segment, which reduces the size of the segment table. The filtering strategies are discussed further in section 4.

The resulting segment tables can be sorted in various ways (any combination of sorting by alphabetical order, length of segments and frequency). It is also possible to strip away the location information, which leaves a list of segments and their corresponding frequency data. The stripped segment list is easier to handle when a translator is viewing or revising the segments. After revision it is possible to extract the location information for each segment in the revised segment list and create a new complete segment table.

3.3 Measuring the recurrence rate

Given a segment table FRASSE can return the recurrence rate for the text.

> **Input:** A segment table with recorded positions
> **Output:** (1) Total number of words in the analysed text; (2) number of words that were part of recurrent segments, and (3) The ratio of (2) to (1) expressed as a percentage.

The algorithm calculates the recurrence rate by positioning all segment types on top of the actual text. The fact that segments may be overlapping is represented in the resulting figures.

3.4 Comparison and combination of several segment tables

It is possible to compare and combine several sets of analysis results (i.e. the intersection or union of different analysis results). This could be useful for large text materials which could be split into subtexts, but where it would be useful to compare the data.

Input: Segment tables, *st1, st2... stn.*
Output: A new segment table which holds the intersection or union of segments in the input tables.

3.5 Performance

The program has been run on texts of various sizes, up to 1 million words. Sentence retrieval is very fast (under a second up to a minute depending on the size of the text), but segment retrieval is considerably slower.

The segment retrieval algorithm has a quadratic time complexity in terms of the number of sentence types and a worst case quadratic complexity in terms of the maximum sentence length. If *m* is the number of sentence types (or segments) in the initial table, the program has to consider *m(m-1)/2* pairs. For a given pair of sentence types with sentences of length *n* and *p*, respectively, the program considers *np* positions as potential starting posititions for a common segment. This happens regardless of the required minimal length of a recurrent segment, since, fo a normal text, only a fraction of the potential starting posititions will require a large amount of processing.

In a test run we doubled the text size and the initial table twice. There were no common segments in the first text and the texts that were subsequently added to it. The test was made on a PC running Windows NT. In the first run a text comprising 25,000 words and 1,765 sentence types was used, which took 45 seconds (for segments of length 2 and larger). In the second run the size was doubled and this time the program needed 3 minutes 15 seconds. The final run made on 100,000 words and 7,060 sentence types took 12 minutes 15 seconds. This agrees with our previous experience of the program and a rough estimate is that 200,000 words of repetitive text takes about 50 minutes to process, 400,000 words 3 hours 15 minutes, etc., given that the program has enough virtual memory available.

When we analysed a legal text of 830,000 words the results were that 32.73 per cent of text were made up by recurrent sentences. The segment analysis yielded that the overall coverage of unrevised recurrent segments made up 73.66 per cent of the full text. The analysis was of course relatively time and processing intensive, so we also divided the text into three subsections which in some sense were related in content and, as expected, the result was that the coverage ratio decreased, but surprisingly little. The coverage ratio varied from 65.81 per cent up to 69.69 per cent for each subtext. The small difference between a full analysis and several partial analyses indicates that a large text does not necessarily have to be analysed in one batch, especially since we have the possibility to compare the results from the partial analyses to find common data.

4. Example from using FRASSE

An example of the segments (length 3 or longer) retrieved by FRASSE is shown below. Note that this list is retrieved without the use of any filters and that it therefore reflects the maximal strings in the source text in a very crude way. The segments could not be fed into a translation memory as they are here; a manual revision or complementing filtering would be necessary.

you want to	452	and then choose	82
, you can	392	the database window	80
for example,	327	in this chapter.	79
menu, choose	136	, click the	79
if you want	130	the edit menu	78
you can use	119	you can also	78
to create a	112	the tool bar	77
, and then	109	a form or	73
example, you	106	in design view	73
if you want to	105	choose the ok button	72
, see chapter	105	you can create	72
for example, you	102	the ok button.	71
in this chapter	100	, select the	71
the qbe grid	99	then choose the	70
form or report	94	choose the ok button.	69
, see "	88	for more information	69

Table 4. The top 32 segments generated by FRASSE from a computer program User's Guide (without filtering)

A problem that arises is what criteria should be used when deciding translation units. When we revised the output from FRASSE by hand we found that a majority of the segments that we removed from the original output actually were segments that ended with function words (such as “and”, "but", "to", "in", etc.) and segments with only one parenthesis character or quotation mark. Therefore we implemented a filter where it is possible to define words that should be stripped at the beginning and at the end of segments as well as requirements on what kinds of characters that should be regarded as pairs (quotation marks, parentheses, etc). In table 5 below, the result is shown when we applied such a filtering mechanism on the same text as in table 4. We used function words and frequent verbs which have a very general meaning in the text type. Of the 32 most frequent segments in the unfiltered result above, 22 have been filtered out, leaving a residue of 10.

in this chapter	100
the qbe grid	99
form or report	94
the database window	80
the edit menu	78
the tool bar	77
the ok button	74
in design view	73
choose the ok button	72
for more information	69

Table 5. The top 10 segments generated by FRASSE (with filtering)

The strategy has a preference to keep noun phrases and prepositional phrases as well as longer segments ending in a non-function word.

4.1 Applications of FRASSE

As we have pointed out earlier the intention of FRASSE is to enhance working with translation memory-based translation tools. The program can also be seen as a diagnostic tool where it is possible for a translation co-ordinator to decide what recurrence profile a given text has. You could compare the document to be translated internally (i.e. by calculating the recurrence rate within the document) or externally (i.e. by comparing the document with related documents that have already been translated and which are stored in existing translation memories). A concrete application for FRASSE would be to construct terminology lists consisting of frequent segments that can be stored in the phrase lexicon of a translation memory tool. It is not necessary to translate them before the translation starts. Instead the translations of the recurrent phrases can easily be made the first time they occur in the source text, making the translation data available the next time the phrase is encountered.

Another application would be in lexicography which is the main aim for systems like XTRACT, mentioned below. One object for a lexicographer is to identify collocations in a text that could be considered as entries in a domain-specific dictionary. As collocations are arbitrary, dependent on the text domain and recurrent, they can in principle only be recovered from corpus material and by performing statistical and linguistic analysis of their occurrence (see Smadja, 1993 for a more elaborate discussion).

Researchers involved in literary analysis would also find analysis of recurrent segments useful when they try to find characteristics of a work of literature. Lessard & Hamm (1991) describes a program that was designed for these purposes. However, they report some shortcomings with their program, e.g. subsegments are not always removed when maximal strings have been identified. Furthermore, their program cannot handle a text as the sole input. First a concordance has to be produced for each lexical item in the text and then this concordance is analysed for repetitive structures. This also meant that they couldn't analyse large texts in a single pass, because the concordance information wouldn't fit into memory.

When companies or institutions produce bulky documentation which requires a team of writers working on the same set of documents, some level of standardisation must be achieved. By analysing already existing material with tools like FRASSE, it would be possible to standardise terminology and phraseology. A segment table from FRASSE sorted alphabetically is an excellent starting point in pin-pointing inconsistent phraseology.

5. Comparisons to related work

Similar systems have been described in other fields of NLP, especially within lexicography and literary analysis (see previous section). We would like to mention the following research and point out the differences with our approach.

5.1 Choueka's n-gram analyser

In Choueka (1988) an n-gram analyser is described which produces collocational expressions of 2 to 6 words. The corpus that was analysed consisted of ten million words of newspaper text. Only sequences of words of length 2 to 6 with a frequency above 10 were retrieved. After the initial analysis the raw list of collocations was filtered in different stages to reduce the size of it. There are two main restrictions in Choueka's work compared to our approach, namely that the maximal segments are not singled out from subsegments and that there is an upper limit on the length of collocational expressions.

5.2 Smadja's XTRACT

XTRACT developed at Columbia university extracts collocations from a parts-of-speech tagged text (Smadja 1993). The program works in several steps, building significant bigrams and expanding these to n-gram collocations. The program retrieves the longest significant collocations, but in order to take full advantage of the algorithm the text that is being analysed must be tagged. In the current version of XTRACT it is not possible to use a text as the only input to the program. You must specify each *seed word* that you want to test for significant bigrams. There are several interesting ideas in Smadja's approach, especially the overall intent to filter out non-significant combinations of words as early as possible in the analysis process in order to increase the efficiency and relevance of the output data.

Apart from Choueka and Smadja, work on identifying collocations in large texts have been made by for example Church & Hanks (1989).

6. Future work

As pointed out earlier in this paper there are several aspects of our approach which we would like to develop further. Here are some examples of what we are working with:

- *Improving the efficiency* of FRASSE. We are confident that the performance of the program could be improved considerably by, for example, using word frequency information to block irrelevant segment processing.
- *Creating a working environment for reviewing translation units.* This entails the possibility to select a segment from a segment table and look at its different contexts in the source text, i.e. a kind of concordance facility.
- *Combining FRASSE with an alignment system* to make it possible to make systematic recurrence analyses of parallel texts.
- *Validating translation units* by running FRASSE on a parallel English-Swedish corpus that we are currently setting up.
- *Retrieval of 'abstract' units* such as phrase patterns, lemmatised segments and conceptual units, e.g. the segment "Press the ESC key" could be analysed as

'Press NP' or "Press <key>". One idea that we would explore further in this context is to use *seed segments* (as a complement to *seed words*) to find relationships between different segments within sentences.

FRASSE is available to the research community via Internet at the time of publication of this paper. Information on how to download the software can be obtained from the authors. The availability of FRASSE will be announced on appropriate discussion lists.

References

Choueka, Y. (1988) "Looking for needles in a haystack". In *RIAO 88, User-oriented Content-based Text and Image Handling,* Volume 1, 609-623, 1988.

Church K. & Hanks, P. (1989). "Word association norms, mutual information and lexicography." In *Proceedings from the 27th Meeting of the ACL*, 76-83.

Language Industry Monitor No. 19 (1994). "Mendez Rolls its Own".

Larsson, A. & Merkel, M. (1994). "Semiotics at Work: Technical Translation and Communication in a Multilingual Corporate Environment". To be published in the *Proceedings of NODALIDA* (Nordiska Datalingvistikdagarna), Stockholm university.

Lessard, G. & Hamm, J.-J. (1991). Computer-Aided Analysis of Repeated Structures: the Case of Stendahl's Armance. In Journal of the Association for Literary and Linguistic Computing., Vol. 6, No. 4, 1991.

Merkel, M. (1992). Recurrent Patterns in Technical Documentation. Research Report LiTH-IDA-R-92-31, Dept. of Computer and Information Science, Linköping University.

Merkel, M. (1993). "When and why should translations be reused?" In *Papers from the XIII VAAKKI symposium 1993*, Vaasa.

Smadja F. (1993) "Retrieving Collocations from Text: Xtract". In *Computational Linguistics*, vol. 19, no. 1.