The *C-value/NC-value* Method of Automatic Recognition for Multi-word Terms

Katerina T. Frantzi[1], Sophia Ananiadou[1], and Junichi Tsujii[2]

[1] Dept. of Computing and Mathematics, Manchester Metropolitan University, Chester Str., Manchester, M1 5GD, U.K.
[2] Dept. of Information Science, University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan

Abstract. Technical terms (henceforth called simply *terms*), are important elements for digital libraries. In this paper we present a domain-independent method for the automatic extraction of multi-word terms, from machine-readable special language corpora.
The method, (*C-value/NC-value*), combines linguistic and statistical information. The first part, *C-value* enhances the common statistical measure of frequency of occurrence for term extraction, making it sensitive to a particular type of multi-word terms, the nested terms. The second part, *NC-value*, gives: 1) a method for the extraction of term context words (words that tend to appear with terms), 2) the incorporation of information from term context words to the extraction of terms.

1 Introduction

Terms, the linguistic representation of concepts [25], are important elements for digital libraries. Rapid changes in many specialised knowledge domains (particularly in areas like computer science, engineering, medicine etc.), means that new terms are being created all the time, making important the automation of their retrieval.

Many techniques for multi-word automatic term recognition (ATR) move lately from using only linguistic information [2,1,3], to incorporating statistical as well. Dagan and Church, [7], Daille et al., [8], and Justeson and Katz, [18], Enguehard and Pantera, [11], use frequency of occurrence. Daille et al., and Lauriston, [21], propose the likelihood ratio for terms consisting of two words. For the same type of terms, Damerau, [9], proposed a measure based on mutual information (MI). Those of the above methods that aim to multi-word terms which may consist of more than two words, use as the only statistical parameter the frequency of occurrence of the candidate term in the corpus. A detailed description and evaluation of previous work on multi-word ATR can be found in [16].

The method we present and evaluate in this paper extracts multi-word terms from English corpora combining linguistic and statistical information. It is divided into two parts: 1) the *C-value*, that aims to improve the extraction of nested multi-word terms [12], and 2) the *NC-value* that incorporates context information to the *C-value* method, aiming to improve multi-word term extraction in

C. Nikolaou, C. Stephanidis (Eds.): ECDL'98, LNCS 1513, pp. 585–604, 1998.

general [15,14]. The first part, *C-value* has been also used for collocation extraction [13]. The second part incorporates a method for the extraction of term context words, which will be also presented and evaluated in this paper.

Since ATR methods are mostly empirical, [19], we evaluate the results of the method in terms of precision and recall, [27]. The results are compared with those produced with the most common statistical technique used for ATR to date, the frequency of occurrence of the candidate term, which was applied on the same corpus.

2 The *C-value* Approach

This section presents the *C-value* approach to multi-word ATR. *C-value* is a domain-independent method for multi-word ATR which aims to improve the extraction of nested terms. The method takes as input an SL corpus and produces a list of candidate multi-word terms. These are ordered by their *termhood*, which we also call *C-value*. The output list is evaluated by a domain expert. Since the candidate terms are ranked according to their termhood, the domain expert can scan the lists starting from the top, and go as far down the list as time/money allow.

The *C-value* approach combines linguistic and statistical information, emphasis being placed on the statistical part. The linguistic information consists of the part-of-speech tagging of the corpus, the linguistic filter constraining the type of terms extracted, and the stop-list. The statistical part combines statistical features of the candidate string, in a form of measure that is also called *C-value*.

Subsections 2.1 and 2.2 describe and justify the linguistic part and the statistical part of the method. Subsection 2.3 describes the algorithm. In subsection 2.4 we apply the method to a medical corpus and present the results. Subsection 2.5 evaluates the results.

2.1 The Linguistic Part

The linguistic part consists of the following:

1. Part-of-speech information from tagging the corpus.
2. The linguistic filter applied to the tagged corpus to exclude those strings not required for extraction.
3. The stop-list.

Tagging.
Part-of-speech tagging is the assignment of a grammatical tag (e.g. noun, adjective, verb, preposition, determiner, etc.) to each word in the corpus. It is needed by the linguistic filter which will only permit specific strings for extraction.

The linguistic filter.
It would be 'very desirable' for a method to be able to extract all types of terms

(e.g. noun phrases, adjectival phrases, verbal phrases, etc.). In such a case the linguistic filter would not be needed. This approach has not yet been followed by us or by any other researchers in ATR. The reason is that the statistical information that is available, without any linguistic filtering, is not enough to produce useful results. Without any linguistic information, undesirable strings such as *of the, is a*, etc., would also be extracted.

Since most terms consist of nouns and adjectives, [26], and sometimes prepositions, [18], we use a linguistic filter that accepts these types of terms.

The choice of the linguistic filter affects the precision and recall of the output list. A number of different filters have been used, [3,8,7,18]. A 'closed' filter which is strict about the strings it permits, will have a positive effect on precision but a negative effect on recall. As an example, consider the filter that Dagan and Church use, [7], the $Noun^+$. This filter only permits sequences of nouns, and as a result produces high precision since noun sequences in an SL corpus are the most likely to be terms. At the same time, it negatively affects recall, since there are many noun compound terms that consist of adjectives and nouns, which are excluded by this filter.

An 'open' filter, one that permits more types of strings, has the opposite effect: negative for precision, positive for recall. An example of such a filter is that of Justeson and Katz, [18]. They extract noun phrases of the form $((Adj|Noun)^+|((Adj|Noun)^*(NounPrep)^?)(Adj|Noun)^*)Noun$. The above filter would extract more terms than the $Noun^+$ one, since terms that contain adjectives and prepositions are also extracted, but it also extracts more non-terms. It extracts terms like *tetracyclines for ocular rosacea, scotomas in low vision, coloboma of retina*, but it also extracts non-terms like *strabismus in children, composition of tears, therapy of strabismus, sensory aspects of strabismus*.

The choice of the linguistic filter depends on how we want to balance precision and recall: preference on precision over recall would probably require a closed filter, while preference on recall would require an open filter.

We are not strict about the choice of a specific linguistic filter, since different applications require different filters. We will present our method combined with each of the 3 filters,

1. $NounNoun^+$,
2. Adj^*Noun^+,
3. $((Adj|Noun)^+|((Adj|Noun)^*(NounPrep)^?)(Adj|Noun)^*)Noun$,

and see how the results are affected. We will also take the results of our method using each of these filters, and compare them with the results from frequency of occurrence when combined with these filters.

The stop-list.
A stop-list for an SL in ATR is a list of words which are not expected to occur as term words in that domain. It is used to avoid the extraction of strings that

are unlikely to be terms, improving the precision of the output list. When used in previous approaches, it is not clear how it is constructed, [6,11]. Our stop-list consists of 229 function and other content words, picked from a sample of our corpus (1/10). The words that are included in the stop-list exhibited high frequencies in that sample of the corpus. Some examples are: *great, numerous, several, year, just, good*, etc.

We should note the fact that because a word has not appeared as a term-word of a specific domain in the past does not guarantee that it will not do so in the future. Consider for example the word *optical*, which is relatively new in computer science. If it were a stop-list word, then terms like, *optical character, optical character recognition, optical character reader, optical laser disc, optical mouse* would have been missed when they first appeared in the domain. The choice of using a stop-list is again a matter of balance between precision and recall. A stop-list benefits precision but could leave out terms that contain 'unexpected' words.

2.2 The Statistical Part

The *C-value* statistical measure assigns a termhood to a candidate string, ranking it in the output list of candidate terms. The measure is built using statistical characteristics of the candidate string. These are:

1. The total frequency of occurrence of the candidate string in the corpus.
2. The frequency of the candidate string as part of other longer candidate terms.
3. The number of these longer candidate terms.
4. The length of the candidate string (in number of words).

We will now examine each of these parameters. The frequency of occurrence of the candidate string in the corpus is, as we have seen, the measure which has been used for multi-word ATR until now. In this case, the termhood of a string equals its frequency of occurrence in the corpus

$$termhood(a) = f(a) \tag{1}$$

where
a is the candidate string,
$f(a)$ its frequency of occurrence in the corpus.

As a statistical measure for ATR, the frequency produces good results since terms tend to occur with relatively high frequencies. For example, in our 800,000 word eye-pathology corpus, *optic nerve* appeared 2,084 times, *Descemet's membrane* 1,666 times, *basal cell carcinoma* 984 times, etc. Of course not all terms exhibit high frequencies: *stromal necrosis, epithelial oedema*, and *congestive glaucoma* appear only 3 times each. Low frequency events cause problems for statistical approaches.

Since frequency produces relatively good results, and since its application to corpora is simple, why are we not satisfied with using just that and look for something more?

Consider the string *soft contact lens*. This is a term in ophthalmology. A method that uses frequency of occurrence would extract it given that it appears frequently enough in the corpus. Its substrings, *soft contact* and *contact lens*, would be also extracted since they would have frequencies at least as high as *soft contact lens* (and they satisfy the linguistic filter used for the extraction of *soft contact lens*). However, *soft contact* is not a term in ophthalmology.

A quick solution to this problem is to extract only a substring of a candidate term if it appears a sufficient number of times by itself in the corpus (i.e. not only as a substring). Then, in order to calculate the termhood of a string, we should subtract from its total frequency its frequency as a substring of longer candidate terms

$$termhood(a) = f(a) - \sum_{b \epsilon T_a} f(b) \tag{2}$$

where
a is the candidate string,
$f(a)$ is its total frequency of occurrence in the corpus,
T_a is the set of candidate terms that contain a,
b is such a candidate term,
$f(b)$ is the frequency of the candidate term b that contains a.
However, the problem is not totally solved. Consider the following two sets of terms from computer science.

real time clock	floating point arithmetic
real time expert system	floating point constant
real time image generation	floating point operation
real time output	floating point routine
real time systems	

Both of these two sets contain *nested terms*. We call *nested terms* those that appear within other longer terms, and may or may not appear by themselves in the corpus. The first set contains the term *real time* and the second the term *floating point*. Except *expert system*, all of the other substrings, *time clock, time expert system, time image generation, image generation, time output, time systems, point arithmetic, point constant, point operation, point routine*, are not terms. So substrings of terms may or may not be terms themselves. Also, terms that are substrings do not have to appear by themselves in a text. As a result, a measure like formula 2 would exclude terms if these have been only found as nested, or if they are not nested but present a very low frequency.
So, could we avoid the extraction of substrings that are not terms, and at the same time extract those substrings that are terms?

Simply by looking at the above two sets of examples, we might suspect that *real time* and *floating point* are terms. The indication is that *real time* appears in every term of the first set, and *floating point* in every term of the second. We have no such indication for *time clock, time expert system, time image generation, image generation, time output, time systems, point arithmetic, point constant, point operation, point routine.*

Because *real time* appears in 5 longer terms, and *floating point* in 4 longer terms, this means that both show 'independence' from the longer terms they appear in. This is not the case for *time clock*, which only appears in one term. The higher the number of longer terms that our string appears as nested in, the more certain we can be about its independence.

The last parameter in the *C-value* measure is the length of the candidate string in terms of number of words. Since it is less probable that a longer string will appear f times in a corpus than a shorter string[3], the fact that a longer string appears f times is more important than that of a shorter string appearing f times. For this reason, we incorporate into the measure the length of the candidate string.

Since the maximum length terms can not be nested in longer terms, and some strings are never found as nested anyway, we distinguish two cases

1. If a is a string of maximum length or has not been found as nested, then its termhood will be the result of its total frequency in the corpus and its length.
2. If a is a string of any other shorter length, then we must consider if it is part of any longer candidate terms. If it appears as part of longer candidate terms, then its termhood will also consider its frequency as a nested string, as well as the number of these longer candidate terms. Though the fact that it appears as part of longer candidate terms affects its termhood negatively, the bigger the number of these candidate terms, the higher would be its independence from these. This latter number moderates the negative effect of the candidate string being nested in longer candidate terms.

The measure of termhood, called *C-value* is given as

$$\textit{C-value}(a) = \begin{cases} log_2|a| \cdot f(a) & \\ & a \textit{ is not nested}, \\ log_2|a|(f(a) - \frac{1}{P(T_a)} \sum_{b \epsilon T_a} f(b)) & \\ & otherwise \end{cases} \tag{3}$$

where
a is the candidate string,
$f(.)$ is its frequency of occurrence in the corpus,
T_a is the set of extracted candidate terms that contain a,
$P(T_a)$ is the number of these candidate terms.

It is obvious that *C-value* is a measure based on the frequency of occurrence of a. The negative effect on the candidate string a being a substring of other longer candidate terms is reflected by the negative sign '–' in front of the $\sum_{b \epsilon T_a} f(b)$. The independence of a from these longer candidate terms is given by $P(T_a)$. That the greater this number the bigger its independence (and the opposite), is reflected by having $P(T_a)$ as the denominator of a negatively signed fraction.

[3] This is based on the assumption that the probability of occurrence of the word a in the corpus is independent from the probability of occurrence of any other word in the corpus, which is not always true, [10].

The positive effect of the length of the candidate string is moderated by the application of the logarithm on it.

2.3 The Algorithm

In this subsection we describe the steps taken in the *C-value* method to construct a list of candidate terms from a corpus.

Step 1
We tag the corpus. As mentioned earlier, we need the tagging process since we will use a linguistic filter to restrict the type of terms to be extracted.

Step 2
This stage extracts those strings that satisfy the linguistic filter and frequency threshold. The terms will be extracted from among these strings. The maximum length of the extracted strings depends on:

1. The working domain. In arts for example, terms tend to be shorter than in science and technology.
2. The type of terms we accept. Terms that only consist of nouns for example, very rarely contain more than 5 or 6 words.

The process of finding this maximum length is as follows: We attempt to extract strings of a specific length. If we do not find any strings of this length, we decrease the number by 1 and make a new attempt. We continue in this way until we find a length for which strings exist.

At this point, extraction of the candidate strings can take place. Initially, a list of strings of each length is created, i.e. a list for the bigrams, a list for the trigrams, etc. Here, we remove the word tag, thereby preventing more than one tag for the same word[4]. The lists contain the strings with their frequency of occurrence.

The lists are then filtered through the stop-list and are concatenated. The longest strings appear at the top, and decrease in size as we move down, with the bigrams being at the bottom. The strings of each length are ordered by their frequency of occurrence.

Step 3
This is the stage where the *C-value* for each of the candidate strings is evaluated. *C-value* is calculated in order of the size of the strings, starting with the longest ones and finishing with the bigrams. The *C-value* for the longest terms is given by the top branch of formula 3.

We set a *C-value* threshold, so that only those strings with *C-value* above this threshold are added onto the list of candidate terms. For the evaluation of *C-value* for any of the shorter strings, we need two more parameters (their frequency as part of longer candidate terms, and the number of these longer

[4] We will provide examples in the next subsection.

candidate terms).
To obtain these two parameters, we perform the following:
For every string a, that it is extracted as a candidate term, we create for each of its substrings b, a triple $(f(b), t(b), c(b))$,
where
$f(b)$ is the total frequency of b in the corpus,
$t(b)$ is the frequency of b as a nested string of candidate terms,
$c(b)$ is the number of these longer candidate terms.
When this triple is first created, $c(b) = 1$ and $t(b)$ equals the frequency of a. Each time b is found after that, $t(b)$ and $c(b)$ are updated, while $f(b)$, its total frequency, does not change.
$c(b)$ and $t(b)$ are updated in the following manner:
$c(b)$ is increased by 1 every time b is found within a longer string a that is extracted as a candidate term.
$t(b)$ is increased by the frequency of the longer candidate term a, $f(a)$, every time b is found as nested. If $n(a)$ is the number of times a has appeared as nested, then $t(b)$ will be increased by $f(a) - n(a)$.

Now in order to calculate *C-value* for a string a which is shorter by one word, we either already have for it a triple $(f(a), t(a), c(a))$ or we do not. If we do not, we calculate the *C-value* from the top branch of formula 3. If we do, we use the bottom branch of formula 3.
In that case, $P(T_a) = c(a)$ and $\sum_{b \epsilon T_a} = t(a)$.

After the calculation of *C-value* for strings of length l finishes we move to the calculation of *C-value* for strings of length $l - 1$. This way it is evident whether the string to be processed has been found nested in longer candidate terms.

At the end of this step, a list of candidate terms has been built. The strings of the list are ranked by their *C-value*.

An example on how *C-value* works can be found in [12].

2.4 The Application on a Medical Corpus

The corpus consists of eye-pathology medical records. Initially we had to delete all the fields with the personal details of each record (i.e. name, address, age, etc.). From each record two fields were kept: the diagnosis and the description of the disease, resulting in a corpus of 810,719 words in upper case. The size is enough for our statistical processing, since it is an SL corpus rather than a GL one. Lehrberger points out that 'lexical restrictions may consist of the exclusion of large parts of the total vocabulary of the language due to restricted subject matter', [22].

The corpus contains orthographical mistakes, e.g. *trabenular* instead of *trabecular*, *meshwrk, meshowrk, mehswrok* instead of *meshwork* etc. It also shows inconsistencies as in the following two cases:

1. The use of hyphens. The same term appears with or without a hyphen or even as one word: *vitreoretinal* and *vitreo-retinal, superonasal,*

supero-nasal, and *supernasal, serofibrinous, sero-fibrinous*, and *sero fibrinous*. The use of a hyphen, as we have seen, is a general problem of term recognition (i.e. variation).

2. The single quotes (‘ and ’). These are used sometimes to enclose a term or even a part of a term. For example *naevoid cells* and *’naevoid cells’, ’naevoid’ cells, V-shaped perforation,* and *’V’-shaped perforation, basaloid cells,* and *’basaloid’ cells*. In most of these cases we removed the single quotes.

We tagged the corpus with Brill’s rule-based part-of-speech tagger, [4,5]. Before tagging, the corpus had to be tokenised following the Penn Treebank Tokenisation, [23]. Punctuation had to be separated from the words, and the corpus placed in a one-sentence-per-line format.

Sample of the corpus before the tokenisation, after the tokenisation but before the tagging, and after the tagging can be found in [16].

The tagged corpus is ready for the extraction of candidate strings that are selected by the linguistic filter, the frequency threshold and the stop-list. In order to check the performance of *C-value* with various filters, we extract 3 lists of candidate strings, using the 3 following filters we mentioned in subsection 2.1. The maximum length strings we extract consist of 5 words.

The frequency threshold used for the 3 lists extracted by those filters, is 3, i.e. only strings with frequency of occurrence of 3 or more are extracted. The stop-list was constructed by examining a sample of the corpus (1/10 of its size) and consists of 229 word.

At this stage, these three lists are those that would be produced using the ‘traditional’ statistical measure for multi-word ATR, i.e. frequency of occurrence plus a linguistic filter. We will use these three lists to compare the performance of the *C-value* list with that of pure frequency of occurrence.

The *C-value* algorithm is applied to each of the three lists. We set the value of the *C-value* threshold 0, i.e. strings with *C-value* greater than 0 will be included in the final list. The strings with a *C-value* of 0 are those found *only* as nested in *one* longer candidate term.

For each of the input lists (i.e. for each of the linguistic filters), one *C-value* list is produced. The strings within each list are ranked according to their *C-value*, ready for evaluation by the domain-expert.

2.5 Evaluation

ATR techniques are mostly based on frequency, since terms tend to appear with high frequencies, [7,18]. *C-value* also uses the parameter of frequency. However, there are terms that appear with very low frequencies: *toxoplasmic chorioretinitis, vernal conjunctivitis, zoster keratitis*, all appear only once. Since *C-value* uses a frequency filter, it will not extract these terms. In order to be able to extract low frequency terms, we should not use a frequency threshold. This is possible[5], but it will increase the manual intervention of the domain expert, who

[5] Then, what *C-value* would aim to do is a re-ranking of the list, moving the real terms closer to the top of the list.

evaluates the produced list to extract the 'real' terms. The list would then be a lot longer: in our corpus, the strings with frequency greater than 2 (in the list of the 2nd linguistic filter) are 2,956. If we include also those with frequency 2, they become 5,560. And if we also include those with frequency 1 the number rises to 16,688. For this reason a frequency threshold is used. If however, the application requires higher recall and permits lower precision, the frequency threshold can be removed, or moved to lower values.

We will now evaluate the results of *C-value* in terms of precision and recall and compare them with those of frequency of occurrence.

There exists a lack of formal or precise rules which would help us to decide between a term and a non-term. Domain experts (who are not linguists or terminologists) do not always agree on termhood. Given this fact, we talk about 'relative' rather than 'absolute' values of precision and recall, in comparison with the alternative proposed method of frequency of occurrence. We will compare the results of the *C-value* method and the method that uses frequency of occurrence, using the three linguistic filters described before.

We calculate precision for each of the three linguistic filters, and compare them with the corresponding result of frequency of occurrence. Since *C-value* is a method to improve the extraction of nested terms, the comparison is made for this category of terms. We also calculate the overall values of precision (over the whole lists).

If we wanted to calculate the absolute value for recall, a domain expert would have had to find all the multi-word terms from the corpus (or a sufficiently large sample of it), and then we would have had to check whether these terms had been extracted by *C-value*. Given the time-consuming nature of this task, we decided to calculate recall with respect to frequency of occurrence, which we used as the baseline method.

Table 1 shows the precision for

1. the candidate terms that have also appeared as nested,
2. the candidate terms that have only appeared as nested,
3. all the candidate terms,

extracted by *C-value* and by frequency of occurrence, using the three linguistic filters. For the first case, the results show that, using *C-value*, precision increases by 6% for the first filter, 7% for the second, and 8% for the third filter. The precision using the third filter is only 1% less than that of the first filter. This shows that with the *C-value* method we can use an open linguistic filter without losing much precision.

For the second case, using the *C-value* method, precision increases by more than 31% for the first filter, 38% for the second, and 31% for the third filter. The precision for the second and third filters are even greater than that of the first. This strengthens the point that with *C-value* we have the freedom to use a more open linguistic filter that extracts more types of terms.

For the third case, the differences are not as impressive as before, due to the fact that there are candidate terms that have never been found as nested, and as such, they are treated by *C-value* in a similar way to frequency (the only

difference being the incorporation of the length of the candidate term). These candidate terms moderate the increase we have on precision for the nested terms when using *C-value*.

		1st filter	2nd filter	3rd filter
also nested	*C-v*	40.76%	44.18%	39.58%
	freq	34.4%	37.59%	31.96%
only nested	*C-v*	50%	60%	54.54%
	freq	18.57%	22%	12.91%
all	*C-v*	38%	36%	31%
	freq	36%	35%	30%

Table 1. Precision: *C-value* vs Frequency

C-value is an additional filter to that of frequency of occurrence, and as such, the maximum recall it can reach is that of frequency of occurrence. Table 2 shows recall compared with frequency of occurrence, for the three linguistic filters. It provides both the overall recall, and the recall for the first 25% of extracted candidate terms. We see that with the *C-value* filter, recall falls less than 2% with the first linguistic filter, and around 2.5% with the second and third linguistic filters. However, regarding the first part of the lists, recall does not fall at all when using the first linguistic filter, and increases by 1% and 1.5% when using the second and third linguistic filters respectively. This shows exactly that *C-value* 'attracts' real terms more than pure frequency of occurrence, placing them closer to the top of the extracted list.

interval	1st filter	2nd filter	3rd filter
overall	98.22%	97.41%	97.47%
first 25%	100%	101.13%	101.41%

Table 2. Recall: *C-value* vs frequency.

3 Incorporating Context Information

In this section we incorporate context information into ATR. Subsection 3.1 provides the rationale for using context information. Subsection 3.2 provides a description of the proposed method to extract term context words and to assign them a weight of 'importance'. The application of the method to our

medical corpus and its evaluation is presented in subsection 3.3. Subsection 3.4 describes *NC-value*, an extension to *C-value* which uses context information for the extraction of multi-word terms. In subsection 3.5 we evaluate the results of *NC-value* on our medical corpus, and compare *NC-value* with *C-value* and frequency of occurrence.

3.1 Context Information

We often use the environment of a word to identify its meaning. In automatic systems the information used for disambiguation is restricted mainly to surface criteria as opposed to semantic, discourse and pragmatic information. Lexical information from the context of words has been used for the construction of thesaurus dictionaries [17]. In that case, the context of a word provides clues to its meaning and its synonyms. Grefenstette's system, SEXTANT, uses local lexical information to acquire synonyms. Words that are used in a lexically similar way are candidates to be synonymous. The nouns, adjectives and verbs from the context of the examined word are used to give hints for its meaning.

Regarding term recognition, Sager, [24], stated that terms are strict about the modifiers they accept:

> "Extended term units are different in type from extended word units in that they cannot be freely modified. There is a very limited range of qualifiers which can be used with the term 'heat transfer'; the word 'heat wave' can be modified by such hyperbolic expressions as 'suffocating' or 'never ending' and a great number of other qualifiers. Extended terms are linguistic representations of essential characteristics whereas in words such collocations are inessential in that they can be omitted without affecting the denotation of the head of the nominal group as a lexeme." [24]

Since extended term units differ from extended word units as far as modification is concerned, we could use information from the modifiers to distinguish between terms and non-terms. Thus, if *consistent* is an adjective that tends to precede terms in medical corpora, and it occurs before a candidate term string, we could exploit this information for the benefit of term recognition. Besides adjectives and nouns, we can expand the use of modifier types to verbs that belong to the environment of the candidate term: the string *show* of the verb *to show* in medical domains is often followed by a term, e.g. *shows a basal cell carcinoma*. The string *called* of the verb *to call*, and the form *known* of the verb *to know*, are often involved in definitions, e.g. *is known as the singular existential quantifier* and *is called the Cartesian product*. We will use the three part-of-speech elements also used by [17] to obtain information about the termhood of a candidate string, when they either precede or follow it. These are

1. nouns (*compound cellular naevus*),
2. adjectives (*blood vessels are present*), and
3. verbs (*composed of basaloid papillae*).

3.2 The Context Weighting Factor

Here we describe a method to create a list of 'important' *term context words* from a set of terms extracted from a specialised corpus. By term context words we mean those that appear in the vicinity of terms in texts. These will be ranked according to their 'importance' when appearing with terms.

The context words we treat are adjectives, nouns and verbs that either precede or follow the candidate term.

The criterion for the extraction of a word as a term context word is *the number of terms it appears with*. The assumption is that the higher this number, the higher the likelihood that the word is 'related' to terms, and that it will occur with other terms in the same corpus. Term context words for a specific domain/corpus are not necessarily the same for another domain/corpus. For this reason, we relate term context words to a specific corpus. For example, the words *present, shows, appear, composed* tend to appear with terms in our medical corpus, but may have different meaning if found in a different domain, e.g. mathematics.
We can express the above criterion more formally with the measure

$$Weight(w) = \frac{t(w)}{n} \tag{4}$$

where
w is the context word (noun, verb or adjective) to be assigned a weight as a term context word,
$Weight(w)$ the assigned weight to the word w,
$t(w)$ the number of terms the word w appears with,
n the total number of terms considered.
The purpose of the denominator n is to express this weight as a probability: the probability that the word w might be a term context word. We will elaborate on this point in the following subsection.

3.3 Evaluation

The context weighting factor is a measure that gives the probability for a context word to appear with terms, by expressing the percentage of terms that the context word has appeared with. This measure is in line with the definition of probability of an event as given in probability theory, [20].

Despite its validity as a probability measure, we believe that it could be strengthened if we (somehow) include information not only from terms but from non-terms as well. In other words, we should consider the number of terms that a candidate context word appears with, as well as the number of non-terms. The second number should negatively affect the degree by which the candidate

context word is a term context word. This parameter has not been incorporated in the current measure.

Let us now consider the type of words that we treat as candidates for term context words. In line with Grefenstette, [17], we use nouns, adjectives and verbs that appear in the candidate term's environment. Our choice is also influenced by Sager, [24], who states that terms are strict in the modifiers they accept. However we believe that further investigation of the following issues may improve the extraction of term context words:

1. Some of the above three lexical categories (nouns, adjectives, verbs) may be more important for termhood information. For example, it could be the case that nouns give more information about the termhood of a candidate term than verbs.
2. Some of the above three lexical categories (nouns, adjectives, verbs) may be more important when they either precede or follow the candidate string. For example, it could be the case that verbs that precede the candidate string are more important than verbs that follow it.

We evaluate the list of the term context words produced by the weighting measure. With this evaluation, we want to establish to what degree the extracted term context words relate to terms. We create a list of context words using the above measure and a set of terms from our corpus. Then, we establish the relationship of the extracted words to

1. another set of terms that does not overlap with the one used to create the list of context words,
2. a set of non-terms.

The words in the list of term context words are ordered by the proposed measure. We will consider three different sets from this list to establish the above mentioned relationship:

1. A set taken from the top of the list.
2. A set taken from the middle of the list.
3. A set taken from the bottom of the list.

Our aim is to establish that the first set shows greater association to terms than the third set, and smaller association to non-terms than the third set. Ideally, the second set's results should be in the middle.

We first take a list of terms from our corpus. We use the list extracted by *C-value*. We extract 200 'real' terms[6] from this list. Using these terms and formula 4 we create the list of term context words.

The 200 verified terms comprise the 20% of terms that have been extracted. We will see how the remaining 80% of terms of the *C-value* list associate with the term context words extracted. We will also see the association of non-terms from the *C-value* list with these term context words.

[6] Terms extracted and verified by a domain expert.

We extract three sets of term context words from the top, the middle and the bottom of the list, each set consisting of 20 words. These sets can be found at [16]. We obtain the number of terms each context word appears with, and we then sum them up obtaining the total number of terms for each set of context words. The percentage of terms over terms plus non-terms for each set is given in table 3. We can see that the top set is associated with 12% more terms than the middle one and 21% more terms than the bottom one.

	top set	middle set	bottom set
terms	56%	44%	35%

Table 3. Context words relating to terms.

Our proposed measure for the extraction of term context words accomplishes its purpose, which is to assign high values to words that tend to appear with terms. We therefore use it to extract term context words that will then be used to improve the extraction of terms.

3.4 NC-Value

In this subsection we present the method we call *NC-value*, which incorporates context information into the *C-value* method for the extraction of multi-word terms. Assuming we have a corpus from which we want to extract the terms, we divide the algorithm into three stages.

First stage
We apply the *C-value* method to the corpus. The output of this process is a list of candidate terms, ordered by their *C-value*.

Second stage
This involves the extraction of the term context words and their weights. These will be used in the third stage to improve the term distribution in the extracted list. In order to extract the term context words, we need a set of terms, as discussed in the previous section. We have chosen to keep the method domain-independent and fully-automatic (until the manual evaluation of the final list of candidate terms by the domain-expert). Therefore, we do not use any external source (e.g. a dictionary) which will provide us with the set of terms to be used for this purpose. We use instead the 'top' candidate terms from the *C-value* list, which present very high precision on real terms. We expect to find non-terms among these candidate terms that could produce 'noise', but these non-terms are scarce enough not to cause any real problems. We have chosen to accept a small amount of noise, i.e. non-terms, for the sake of full automation. These 'top'

terms produce a list of term context words and assign to each of them a weight following the process described in the previous section.

Third stage
This involves the incorporation of context information acquired from the second stage of the extraction of multi-word terms. The *C-value* list of candidate terms extracted during stage one is re-ranked using context information, so that the real terms appear closer to the top of the list than they did before, i.e. the concentration of real terms at the top of the list increases while the concentration of those at the bottom decreases. The re-ranking takes place in the following way: Each candidate term from the *C-value* list appears in the corpus with a set of context words. From these context words, we retain the nouns, adjectives and verbs for each candidate term. These words may or may not have been met before, during the second stage of the creation of the list with the term context words. In the case where they have been met, they retain their assigned weight. Otherwise, they are assigned zero weight. For each candidate term, we obtain the context factor by summing up: the weights for its term context words, multiplied by their frequency appearing with this candidate term.
For example, assume that the candidate word W appears 10 times with the context word c_1, 20 times with the context word c_2, and 30 times with the context word c_3. Assume also that the weight for c_1 is w_1, the weight for c_2 is w_2, and the weight for c_3 is w_3. Then, the context factor for W is:
$10 \cdot w_1 + 20 \cdot w_2 + 30 \cdot w_3$

The above description is the second factor of the *NC-value* measure which re-ranks the *C-value* list of candidate terms. The first factor is the *C-value* of the candidate terms. The whole *NC-value* measure is formally described as

$$NC\text{-}value(a) = 0.8\,C\text{-}value\ (a) + 0.2 \sum_{b \epsilon C_a} f_a(b) weight(b) \tag{5}$$

where
a is the candidate term,
C_a is the set of distinct context words of a,
b is a word from C_a,
$f_a(b)$ is the frequency of b as a term context word of a,
$weight(b)$ is the weight of b as a term context word.

The two factors of *NC-value*, i.e. *C-value* and the context information factor, have been assigned the weights 0.8 and 0.2 respectively. These have been chosen among others after experiments and comparisons of the results, as we will discuss in the following section.

3.5 Evaluation

The top of the list produced by *C-value* is used for the extraction of term context words, and the list produced by *C-value* is re-ranked by *NC-value*. However, *NC-value* can be viewed independently from the *C-value* in the sense that in the above

sentence we can substitute *C-value* with a different method for the extraction of terms. That is, the proposed method for incorporating context information can be applied to other approaches for term extraction, i.e. frequency of occurrence.

Let us now consider the creation of the list with the term context words, to be used by *NC-value*. The top candidate terms from the *C-value* list are used[7], since these show high precision on real terms. It is expected that among those terms there will be some non-terms as well. This is unavoidable since we have chosen to keep this process fully-automatic. Full-automation can be sacrificed for the sake of 'correctness' in different applications. In that case, a domain expert would have to check the top of the *C-value* list, that will be used for the extraction of the term context words, and remove the non-terms. The process after this would remain the same.

Regarding the weights 0.8 and 0.2 that have been assigned to *C-value* and the context factor in the *NC-value* measure, these were chosen among others after a series of experiments. The combination 0.8–0.2 gave the best distribution in the precision of extracted terms.

Regarding the evaluation of the results, we carried out tests using the *C-value* list produced by the linguistic filter which includes the preposition 'of'. We chose this filter as it was the most open among the three used, and as such it was the most flexible, accommodating many domains.

The *NC-value* list can be found in [16]. The evaluation will be in terms of precision only. The recall of the *NC-value* list is the same to that of the *C-value* list, since *NC-value* re-ranks the *C-value* list without adding or deleting any candidate terms. As such, the recall of the *NC-value* list is 97.47% (with respect to the real terms extracted by the method based on frequency of occurrence).

The overall precision is the same for the *C-value* list, i.e. 31%. What is different is the distribution of terms in the extracted list. Table 4, and table 5, show the precision and accumulative precision of the *NC-value* list, in comparison with the corresponding *C-value* and frequency of occurrence for the intervals of the ordered candidate terms in the lists. The intervals have been chosen so as to have approximately the same number of n-grams among the lists of the three methods.

The first column in table 4 shows the three methods used. The remaining columns show the precision for each method within the specified intervals. For example, the precision of the *NC-value* for the first interval [top to 40] is 75.70%. The same format is used in table 5, where the accumulative precision is presented.

From the above, we observe that *NC-value* increases the concentration of real terms at the top of the list. More precisely, we observe that *NC-value* brings a 5% increase in precision for the first two intervals.

For the third interval we see a small drop in precision, which is even smaller for the fourth interval. These drops are expected and are desirable due to the increase of precision for the first two intervals. The drops seem smaller than the increases for the first two intervals just because the third and fourth intervals contain a large number of strings.

[7] The first 5% extracted candidate terms were used for these experiments.

	[top–40]	(40–10]	(10–4)	[4–bottom]
NC-value	75.70%	36.08%	26.41%	25.60%
C-value	70.84%	31.31%	27.11%	25.56%
frequency	69.62%	31.64%	24.94%	25.44%

Table 4. Precision: *NC-value*, *C-value* and frequency.

	[top–40]	(top–10]	(top–4)	[top–bottom]
NC-v	75.70%	46.14%	32.80%	31.15%
C-v	70.84%	42.24%	33.04%	31.15%
freq	69.24%	41.70%	33.50%	29.70%

Table 5. Accumulative precision: *NC-value*, *C-value* and frequency.

4 Conclusions

This paper presented and evaluated the *C-value/NC-value* domain-independent method for the semi-automatic extraction of multi-word terms from special language English corpora. We showed two main points:

1. Using more statistical information than the pure frequency of occurrence of candidate terms, improves the precision of the extracted nested multi-word terms, with a slight only loss on recall.
2. Using information from the context of the candidate terms, improves their distribution in the extracted list, i.e. real terms tend to appear closer to the top, while non-terms concentrate closer to the bottom of the list.

We note here that this work was tested in only one corpus. This corpus consisted of medical records and belongs to a specific text type that covers well-structured texts. Although we have shown that the method performs well for this text type of corpora, we are cautious in making this claim for other types of special language corpora, before conducting appropriate experiments.

5 Acknowledgments

We thank Dr. Tom Sharp for providing us with the corpus, and Dr. Michael Florin and Az Bakar for evaluating the extracted lists of candidate terms.

References

1. Ananiadou, S.: A Methodology for Automatic Term Recognition. Proceedings of the 15th International Conference on Computational Linguistics, COLING'94, (1994) 1034–1038
2. Ananiadou, S.: Towards a Methodology for Automatic Term Recognition. University of Manchester Institute of Science and Technology (1988)
3. Bourigault, D.: Surface Grammatical Analysis for the Extraction of Terminological Noun Phrases. Proceedings of the 14th International Conference on Computational Lingustics, COLING'92, (1992) 977–981
4. Brill, E.: A simple rule-based part of speech tagger. Proceedings of the 3rd Conference of Applied Natural Language Processing, ANLP'92, (1992)
5. Brill, E.: A Corpus-Based Approach to Language Learning. Ph.D. Thesis, Dept. of Computer and information Science, University of Pennsylvania (1993)
6. Dagan, I., Pereira, F., Lee, L.: Similarity-Based Estimation of Word Cooccurence Probabilities. Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, ACL'94, (1994) 272–278
7. Dagan, I., Church, K.: Termight: Identifying and Translating Technical Terminology. Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics, EACL'95, (1995) 34–40
8. Daille, B., Gaussier, E., Langé, J.: Towards Automatic extraction of Monolingual and Bilingual Terminology. Proceedings of the 15th International Conference on Computational Linguistics, COLING'94, (1994) 515–521
9. Damerau, F.J.: Generating and Evaluating Domain-Oriented Multi-Word Terms from Texts. Information Processing & Management **29** (1993) 433–447
10. Dunning, T.: Accurate Methods for the Statistics of Surprise and Coincidence. Computational Linguistics **19** (1993) 61–74
11. Enguehard, C., Pantera, L.: Automatic Natural Acquisition of a Terminology. Journal of Quantitative Linguistics **2** (1994) 27–32
12. Frantzi, K.T., and Sophia Ananiadou, S., Tsujii, J.: Extracting Terminological Expressions. The Special Interest Group Notes of Information Processing Society of Japan, 96-NL-112, (1996) 83–88
13. Frantzi, K.T., Ananiadou, S.: Extracting Nested Collocations. Proceedings of the 16th International Conference on Computational Linguistics, COLING'96, (1996) 41–46
14. Frantzi, K.T., Ananiadou, S., Tsujii, J.: Automatic Term Recognition using Contextual Cues. Proceedings of the 2nd Workshop on Multilinguality in Software Industry (MULSAIC'97), 15th International Joint Conference on Artificial Intelligence, IJCAI'97, (1997) 73–79
15. Frantzi, K.T.: Incorporating Context Information for the Extraction of Terms. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL) and 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL), (1997) 501–503
16. Frantzi, K.T.: Automatic Recognition of Multi-Word Terms. Ph.D. Thesis, Manchester Metropolitan University Dept. Of Computing & Mathematics, in collaboration with UMIST Centre for Computational Linguistics, (1998)
17. Grefenstette, G.: Explorations in Automatic Thesaurus Discovery. Kluwer Academic Publishers, (1994)
18. Justeson, J.S., Katz, S.M.: Technical terminology: some linguistic properties and an algorithm for identification in text. Natural Language Engineering **1** (1995) 9–27

19. Kageura, K., Umino, B,: Methods of Automatic Term Recognition -A Review-. Terminology **3** (1996) 259–289
20. Larson, H.J., Larson, J.: Introduction to probability theory and statistical inference. Wiley series in probability and mathematical statistics, Wiley, New York, Chichester (1982)
21. Lauriston, A.: Automatic Term Recognition: performance of Linguistic and Statistical Techniques. Ph.D. Thesis, University of Manchester Institute of Science and Technology (1996)
22. Lehrberger, J.: Sublanguage analysis. Analyzing language in restricted domains, Ralph Grishman and Richard Kittredge (editors), Lawrence Erlbaum, **2** (1986) 19–38
23. Penn: Penn Treebank Annotation. Computational Linguistics **19** (1993)
24. Sager, J.C.: Commentary by Prof. Juan Carlos Sager, Actes Table Ronde sur les Problèmes du Découpage du Terms, Montréal, 26 aouũt. Guy Rondeau, AILA–Comterm,Office de la Langue Francaise, Québec, (1978) 39–74
25. Sager, J.C., Dungworth, D., McDonald, P.F.: English Special Languages: principles and practice in science and technology. Oscar Brandstetter Verlag KG, Wiesbaden, (1980)
26. Sager, J.C.: A Practical Course in Terminology Processing. John Benjamins Publishing Company, (1990)
27. Salton, G.: Introduction to modern information retrieval. Computer Science, McGraw-Hill (1983)