N-GRAM CLUSTER IDENTIFICATION DURING EMPIRICAL KNOWLEDGE REPRESENTATION GENERATION

Robin Collier

Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield, S1 4DP, England
r.collier@dcs.shef.ac.uk

Abstract:

This paper presents an overview of current research concerning knowledge extraction from technical texts. In particular, the use of empirical techniques during the identification and generation of a semantic representation is considered. A key step is the discovery of *useful* n-grams and correlations between clusters of these n-grams.

keywords: knowledge representation, large text corpora, language understanding.

1. BACKGROUND

The primary knowledge extraction and text retrieval conferences (MUC-4, 1992; TREC-1, 1993; TIPSTER, forthcoming) utilise domain-specific queries and templates to identify relevant concepts from within a corpus and extract applicable documents or information.

The structures generated by the system discussed in this paper are similar to these domain-specific templates, they could be used for compact representation of information contained in documents for text retrieval purposes. The automatic generation of templates would be a significant development.

The motivation for generating a domain specific representation is similar to that of Riloff (1993), although the approach is quite different. The conceptual sentence analyser developed at the University of Massachusetts, CIRCUS (Lehnert, 1990), contains a part-of-speech lexicon and a manually constructed concept dictionary. Riloff's AutoSlog will automatically generate a domain-specific concept dictionary.

Case frames are used to represent concepts. Each concept contains a range of information. The *trigger* is a specific word or phrase identifying a potential match. A set of *enabling conditions* defines constraints that require satisfaction. Relevant information, extracted from the surrounding context, is placed into *variable slots* which define information such as objects and actors. Each *variable slot* has a *syntactic expectation* associated with it which defines the expected linguistic context. Slot *constraints* define selectional restrictions on the slot filler. Finally, information that is common to all instantiations of the concept is defined in *constant slots*.

Autoslog utilises a set of heuristics to determine which words and phrases are likely to activate useful concept nodes. For example, the *conceptual anchor point heuristics* define typical linguistic contexts surrounding prospective *triggers*.

A variety of other systems which generate domain-specific representations are discussed in a survey paper by Collier (forthcoming). Some of these systems generate structures that are similar to templates, for example GENESIS (Mooney, 1985), and others acquire domain specific semantic representations, for example MAIMRA (Siskind, 1990).

2. SYSTEM OVERVIEW

The approach acquires a domain specific semantic representation by carrying out stochastic analysis on a large corpus from a technical domain. High frequency phrases are identified and used to recognise groups of paragraphs containing similar subsets of these phrases. It is assumed that, in general, the similarities between paragraphs within each group will define stereotypical concepts. Tools will enable a domain expert to view and manipulate these sets of paragraphs and generate a hierarchical semantic representation of concepts.

The corpus and semantic representation are used to generate schematic structures within the technical domain. Each structure consists of a list of semantic concepts. Sets of structures which have a high level of correspondence are generated. It is assumed that stereotypical structures are represented by similarities between the members of sets containing a sufficient number of structures, and sufficient correspondence. These are stored in a structure knowledge base.

The structures represent stereotypical situations such as lists of actions (e.g. scientific experiments), and common textual information (e.g. the definition of application areas). They are used to translate the existing texts into a semantic/pragmatic representation and store the knowledge in a concise and structured format in a technical knowledge base.

New texts are processed immediately after publication, dynamically updating the technical knowledge base. If segments of new texts cannot be processed by the existing structures, then they are analysed and a novel structure is appended to the structure base.

Collier (1993) presents a more comprehensive outline of the system's architecture and some preliminary stochastic analysis.

3. PARAGRAPH CLUSTERING

The fundamental stage in the process described above is the generation of a domain specific semantic representation. The approach identifies clusters of *useful* n-grams

within paragraphs which correlate with other paragraphs. The term *useful* defines n-grams that have certain qualities, such as a high frequency of occurrence, and a wide distribution over texts within the domain.

There are two principal steps in the identification of these clusters: to recognise useful n-grams of varying lengths within a corpus, and to recognise sets of paragraphs which contain similar clusters, and therefore correlate.

3.1 Structures

Five fundamental structures are used during the identification of correlating paragraphs.

3.1.1 Unique word/integer array

The first structure is an associative array containing an entry for each unique word in the corpus. Each entry is indexed by the word, and holds a unique integer representing that word.

This array is used to translate the textual corpus into a list of integers. All subsequent processing is carried out on this list of integers, this increases efficiency.

The remaining four structures have the same format. Rather than being in word order, as the original text is, identical words are grouped together in the array. These word groups are ordered according to their size. For this reason, the word with the highest frequency of occurrence within the text will exist at the beginning of the array. Figure 1 gives an example of the typical array format.

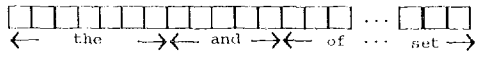

Fig. 1: array format

The highest frequency word that occurs within the text is *the*, therefore its group is at the beginning of the array. The second highest frequency word is *and*, then *of*, etc. The lowest frequency word is *set*, its group is positioned at the end of the array.

The information contained in each of the remaining four arrays is explained below.

3.1.2 Word order array

Due to the grouping of words, the word order will have been lost. The second structure defines this, it contains pointers to the next word in the text. Figure 2 shows the positions of the pointers representing the phrase "... the set of ...".

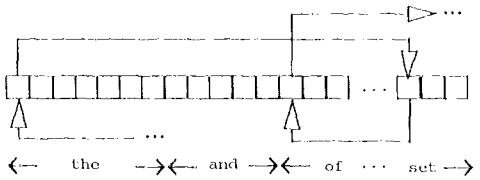

Fig. 2: word order array

3.1.3 Next word array

The third structure contains the unique integer representing the next word pointed to in the text. The value of this will be the integer that represents the word group which the word ordering array element points to.

It is clear that the grouping of the words in the arrays makes it necessary to create additional arrays and complicates the existing ones. The advantage of this grouping is increased computational efficiency.

An example of the enhanced efficiency can be demonstrated by considering the identification of similar n-grams. The next word array groups together next word values which are present after identical words in the text. For example, if the two word phrases *the book*, *the car*, *the book* and *the explosion* were present in the text, then integers representing *book*, *car*, *book* and *explosion* would be grouped together in the next word array. When testing for similar n-grams it is only necessary to look through one section of the array to identify sets of identical n+1-grams, rather than it being necessary to jump to many different positions within an extremely large array. This increases the efficiency of memory access due to the enormous reduction in memory paging.

3.1.4 Phrase length array

The fourth structure contains a phrase length associated with each word. For example, a 1 represents an individual word, 2 represents a bi-gram (the word and the one that is pointed to as the next one), etc.

After the process is complete this array will associate the useful n-grams with their initial word and also define their length.

3.1.5 Next phrase array

The final structure is related to the fourth. Each corresponding entry is a pointer to the next identical phrase. For example, if there were three occurrences of *the set of numbers* in the corpus, then there would be three entries in the phrase length array containing a 4. Each of the corresponding entries in the next phrase array would point to the next identical phrase (figure 3).

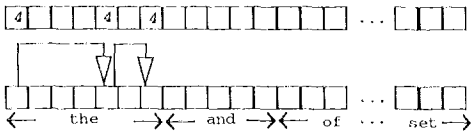

Fig. 3: phrase length and next phrase arrays

3.2 Algorithm

The two principal steps of the process described in section 3 can be divided into six substeps. The first four substeps represent the identification of useful n-grams of varying lengths within a corpus, and the last two represent the identification of sets of paragraphs which contain similar clusters.

Each of the substeps, which create and manipulate the structures defined in section 3.1, is explained below.

3.2.1 Word/integer generation

This procedure produces three arrays. The first associates each unique word with a unique integer, the second defines the frequency of occurrence of each word, and the third contains pointers to the first position of each word group in the array format defined in figure 1.

Initially, each word in the corpus is read sequentially. If an entry associating the word with a unique integer isn't already present, then one is created. If an entry is present, another array containing the frequency of occurrence of each word is incremented.

The array containing the words and their associated unique integer is sorted into descending order by considering each word's frequency. Therefore the highest frequency word is associated with 1, the second highest with 2, etc. An array is also created which contains the initial index positions of each unique word in the word grouping format (figure 1). For example, the highest frequency word would have an initial index of zero. If it had a frequency of 10, then the second highest frequency word index would be 10. If the second word had a frequency of eight, then the third highest frequency word's index would be 18. This indexing array is required during the creation of the word order and next word arrays.

3.2.2 Integer translation

This stage creates three arrays. The first and second are the word order and next word arrays, defined in sections 3.1.2 and 3.1.3. The third is an array associating each document in the corpus with the position, in the word order array, of its first word.

This procedure sequentially processes each word of each document. As each new document commences, the document name and the pointer value associated with the first word are stored in an array. This enables the beginning of any document to be accessed.

For each word, the associated index position from the array generated in the previous step is looked up. This index value is stored in the position in the word ordering array of the previous word that was read. Therefore, defining that this is the index of the next word after the previous one. It also stores the current word's unique integer in the position in the next word array of the previous word that was read. Therefore, defining that this is the unique integer of the next word after the previous one. Finally, it increments the index pointer of the word, as this position has now been filled.

At the end of each paragraph a special integer representing the carriage return is placed in the next word array, this enables identification of paragraph boundaries.

3.2.3 Generate phrase lengths

This step generates three arrays. The first is the phrase length array defined in section 3.1.4. The second is the next phrase array defined in section 3.1.5. The third is similar to the previous one, but it points to the previous identical phrase rather than the next. The algorithm becomes rather complicated when overwriting existing entries in the second and third arrays. This is due to the manipulation of the pointers to the next and previous identical phrases.

Each of the groups of similar words are processed in turn (e.g. in figure 1 all of the *the*'s, then the *and*'s, etc.). The next word array is used to identify the word following the first *the* in the group. Then all of the other *the*'s are checked to identify those with the same next word, creating a set of those that match. This set represents all of the phrases within the corpus that are the same as the first bi-gram.

The phrase length is incremented to two, and this matching process is repeated for the next word of the original phrase (i.e. the third word of the n-gram), but only on the reduced set of previously matching words. This process continues until the longest phrase which occurs a multiple number of times is generated, or a carriage return is encountered.

If the final phrase length is greater than one, then each of the words in the matching set is processed in turn. If the position pointed to by the word does not already have a phrase associated with it, then the phrase length is stored in the associated position in the phrase length array. The position in the next phrase array of the previous phrase in the matching set is updated with the current phrase's position, and therefore defines that this is the next identical phrase after the previous one. Also, an array which defines the previous phrase's position is updated by storing the pointer value of the previous phrase in the current phrase's slot, and therefore pointing to the previous identical phrase.

If the position does already contain a phrase length that is longer, then the current phrase is missed out and the next one processed. In this case the position already has a longer n-gram associated with it.

If the new phrase length is longer than the current one, then the phrase is overwritten, but the pointers to the previous and next phrase require updating. For example, if both a previous phrase and a next phrase pointer exist then the current position should be removed from the linking up of the existing set of identical phrases (figure 4). It is necessary to alter the next phrase value of the previous phrase (which is currently set to the position to be overwritten) to the current positions next phrase. Also the next phrase's previous phrase position (which holds the current position to be overwritten) requires updating to the current phrases previous phrase position.

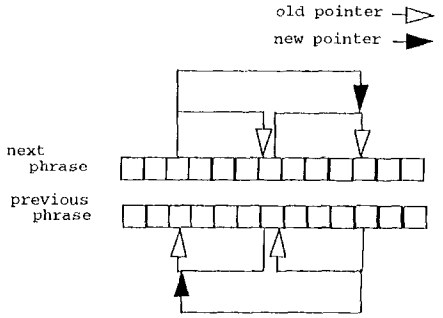

Fig. 4: identical phrase removal

This process is repeated for all of the other *the*'s in turn, and then for each of the other groups, generating the longest n-grams which have at least two occurrences.

3.2.4 Identify useful n-grams

The fourth step is the identification of the n-grams that provide effective correlations between phrases and paragraphs. The phrase length and next phrase arrays are revised so that they only contain these n-grams.

The previous process will have identified the longest phrase that occurs a multiple number of times in the corpus. The phrase length array is traversed and each phrase with this longest length is stored in a set. At the same time, the next phrase array is used to identify the frequency of occurrence of each phrase. This can be obtained by counting while traversing through the pointers to the next identical phrase.

This set of longest phrases is arranged in ascending order by frequency of occurrence. The n-best remain in the phrase length and next phrase arrays. The value of n will depend on the domain being analysed. A domain with considerable correlation will have a greater n than a domain with little correlation. This is an area for further investigation after development of the entire system.

All of the subphrases that exist within these n-best are deleted from the arrays. For example in the phrase *the set of numbers*, subphrases *set of numbers* and *of numbers* will be deleted and so that they are not considered during further analysis.

Those that do not exist within the n-best have their associated phrase lengths reduced by one. This shorter phrase is compared with all other phrases of the same length in the group to identify whether it is identical to an existing phrase. If this is the case, then the next phrase pointer of the last phrase in the set will be altered to point to the first phrase in the identical phrase set, and vice-versa for the previous phrase array.

This entire process is repeated, reducing the length of the phrases to be considered by one each time. Therefore, the second iteration will consider phrases with a length equal to the longest phrase minus one, the third iteration considers phrases with a length equal to the longest phrase minus two, etc.

When this process is complete the phrase length and next phrase arrays will contain all of the *useful* phrases.

The final two processes identify clusters of phrases within individual paragraphs which correlate with clusters of phrases in other paragraphs.

3.2.5 Paragraph weight parse

This procedure associates each paragraph with a weight representing its probability of correlating with other paragraphs. The weight considers factors such as the size of the paragraph, the size and frequency of n-grams existing within that paragraph, and the distribution of the n-grams throughout the corpus.

The actual process is relatively straightforward. The corpus is parsed, beginning at the first word and using the pointers in the next word array. This will traverse the words in the order of the original text, enabling identification of all n-grams in each paragraph and using them in an equation to assign the correlation weight.

The current equation to generate paragraph weights is:

$$\frac{(\text{num bi-gs}*2.5)+(\text{num tri-gs}*4)+..+(\text{num n-gs}*(n+((n-1)*0.5))}{\text{total no words in paragraph}}$$

This equation is simple but accounts for all the important factors listed above, apart from the distribution of the n-grams within the corpus.

These weights are used to sort the paragraphs into ascending order.

3.2.6 Identify useful paragraph clusters

The final process identifies all of the sets of correlating paragraphs within the corpus, and extracts the highest quality correlations.

Each paragraph produced in the previous step is processed in turn. Using the next phrase array, all paragraphs which correlate with at least one n-gram are identified.

Groups of paragraphs containing identical subsets of n-grams are identified and placed into sets. Each of these sets can then be assigned a weight representing the quantity, i.e. number of paragraphs, and quality, i.e. number and size of n-grams.

The final step is to sort the correlation weights into ascending order.

The system has now produced a list of n-gram clusters representing paragraph correlations. These are ordered by considering the quality of n-grams within the cluster, and the quantity of correlation occurring with other paragraphs. From the assumptions outlined in section 2, "the similarities between paragraphs within each group will define stereotypical concepts", these clusters will be extremely useful in the generation of a domain specific semantic representation.

4. PRELIMINARY RESULTS

The entire system, which is discussed in section 2, is currently under development. The stage concerning the identification of correlating paragraphs, which is discussed in section 3, has only recently been implemented. For this reason there are a limited number of results to report upon.

The corpus currently being considered consists of 82 chemical patents containing over half a million words. The programs are being run on a Sun™ Sparcstation Classic with 32 megabytes of RAM.

Table 1 presents an elementary example which is intended to demonstrate the systems scope for improvement as larger corpora are considered. It shows that it is possible to identify paragraphs which sufficiently correlate to provide a strong indication of fundamental concepts within the domain. In this example, a common stage of an experiment is being indicated.

The results in table 1 were gained from analysis of a single patent containing approximately 14000 words. In the patent, 15 paragraphs contained the 4 gram (this is defined by a **4** after the first word of the n-gram) *This was prepared from*, and two of these contained the 9-gram *oxime and 3-methoxycarbonyl-1-vinyloxy-carbonyl-1,2,5,6-tetrahydropyridine and recrystallised from methanol/diethyl ether, mp.*

This **4 was prepared from** isopropyl carboxamide **oxime **9** and 3-methoxycarbonyl-1-vinyloxy-carbonyl-1,2,5,6-tetrahydropyridine and recrystallised from methanol/diethyl ether, mp** 112øC, Rf = 0.28 in dichloromethane/methanol (20:1) on silica.

This **4 was prepared from** phenylacetamide **oxime **9** and 3-methoxycarbonyl-1-vinyloxy-carbonyl-1,2,5,6-tetrahydropyridine and recrystallised from methanol/diethyl ether, mp** 154-158øC, Rf = 0.63 in dichloro-methane/methanol (20:1) on alumina.

Table 1: examples of paragraph correlation

Further correlations exists between the two paragraphs which have not been identified by the system due to the n-grams either being small or containing minor textual differences (e.g. *øC*, *Rf* =, and *dichloro-methane/methanol (20:1) on*).

Many more examples can be drawn from the analysis of this single patent which contain a large number of correlating n-grams but are too large and complicated to report on in this paper.

Finally, an interesting result was that a 69-gram was identified which occurred twice within the single patent. It concerned the explanation of a diagram presenting the structure of a compound.

5. CONCLUSIONS

I am not aware of any techniques, within knowledge representation generation research, which are significantly similar to this clustering approach. The novelty is due to the use of n-gram correspondences during the identification of sets of paragraphs containing similar conceptual information, and the employment of these examples to emphasise the fundamental concepts within the domain. For this reason, it could prove to be a rewarding area for further research.

Due to the nature of technical documents and technical language, a large quantity of the phrases used are highly structured and standardised. This formalism implies that the n-gram clustering approach will produce effective results during the identification of conceptually similar paragraphs.

An essential test will be the assessment of a domain-specific semantic representation created using the correlating paragraphs generated by the system and the tools mentioned in section 2. It will be necessary to evaluate the scope and quality of the representation. One possibility is to compare, using an identical corpus, a representation created by a group of experts with that of the system.

The fundamental point to convey is that as larger corpora are analysed the quantity of examples and quality of correlations will improve. The results of further experimentation and analysis will be reported in future publications.

Although this knowledge representation generation is the fundamental stage of the process outlined in section 2, it is only a fragment of the entire system. An application developed using this process has the potential to be invaluable for domain specialists who wish to identify documents containing similar conceptual information within extremely large knowledge bases.

6. REFERENCES

Collier, R. (1993). Knowledge acquisition from technical texts using natural language processing techniques. *Proceedings of the 2nd Workshop on the Cognitive Science of Natural Language Processing*, pp. 11.1 to 11.15. Dublin, Eire: Dublin City University.

Collier, R. (forthcoming). An historical overview of natural language processing systems that learn. *Artificial Intelligence Review.* Kluwer Academic Publisher: Dordrecht, Germany.

Lehnert, W. (1990). Symbolic/subsymbolic sentence analysis: exploiting the best of two worlds. In Barnden, J. and J. Pollack (Eds.), *Advances in Connectionist and Neural Computation Theory*, volume 1, pp. 135 to 164. Ablex Publishers: Norwood, NJ.

Mooney, R. (1985) Generalising explanations of narratives into schemata. *Technical Report T-147.* Co-ordinated Science Laboratory, University of Illinois, Urbana.

MUC-4 (1992). *Proceedings of the Fourth Message Understanding Conference.* Morgan Kaufmann: San Mateo, CA.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. *Proceedings of the Eleventh National Conference of Artificial Intelligence.* Washington, D.C.: MIT Press, Cambridge, MA.

Siskind, J.M. (1990) Acquiring core meanings of words, represented as Jackendoff-style conceptual structures, from correlated streams of linguistic and non-linguistic input. *Proceedings of the Twenty-eighth Annual Meeting of the Association for Computational Linguistics*, pp. 143 to 156. University of Pittsburgh, Pennsylvania: Association for Computational Linguistics.

TIPSTER (forthcoming). *Proceedings of TIPSTER Text Phase I, 24 Month Conference.* Morgan Kaufmann: Fredericksburg, Virginia.

TREC-1 (1993). *Proceedings of The First Text Retrieval Conference,* Harman, D.K. (Ed.). National Institute of Standards and Technology: Gaithersburg, Maryland.