Efficient Mining of Textual Associations

Alexandre Gil and Gaël Dias

Beira Interior University, Computer Science Department,
6200-053 Covilhã, Portugal
agil@pt.ibm.com, ddg@di.ubi.pt

ABSTRACT

This paper describes an efficient implementation for mining textual associations from text corpora. In order to tackle real world applications, efficient algorithms and data structures are needed to manage, in reasonable time and space, the overgrowing volume of text data. For that purpose, we introduce a global architecture based on masks, suffix arrays and multidimensional arrays to implement the SENTA extractor [Dias, 2002]. In particular, SENTA has shown great flexibility and accuracy to mine textual associations such as collocations, cognates, morphemes and chunks. Our solution shows $O(h(F)\ N \log N)$ time complexity and $O(N)$ space complexity where N is the size of the corpus and $h(F)$ is a function of the context window size.

Keywords: Suffix Arrays, Masks, Text Mining.

1. INTRODUCTION

Text mining applications tend to infer knowledge whether about the language itself or about the information transmitted by the texts. Our work focuses on the extraction of relevant textual patterns that embody knowledge about the language. In this context, word collocations, local patterns of syntactical sequences and relevant character associations may be automatically extracted from text corpora.

In the context of word associations, collocations (sequences of words that co-occur more often than expected by chance) are frequently used in everyday language, usually to precisely express ideas and concepts that cannot be compressed into a single word. For instance, [Bill of Rights], [swimming pool], [as well as], [in order to], [to comply with] or [to put forward] are collocations. As a consequence, their identification is a crucial issue for applications that require a certain degree of semantic processing (e.g. machine translation, information extraction, information retrieval or summarization).

But, textual associations are not restricted to word associations. Indeed, [Argamon-Engelson *et al.*, 1998] assess that the identification of local patterns of syntactical sequences is essential for various applications areas including word sense disambiguation, bilingual alignment and text summarization. Indeed, relevant syntactical sequences such as [AT JJ NN][1], [JJ NP CC JJ NP], [NP $ JJ NN], [NP CO NP CO NP CC NP] or [HV RB BEN] embody relevant structures that allow shallow sentence understanding.

Finally, in the context of character associations, the decomposition of words into morphemes or cognates has proved to lead to improved results in different

[1] We use the Brown part-of-speech tag set: AT = determinant, JJ = adjective, RB = adverb, NN = singular noun, NP = personal noun, $ = possessive markup ('s), CO = comma, CC = coordination conjunction, HV = auxiliary have, BEN = past participle of verb to be.

areas including text indexing, bilingual alignment and information extraction. Indeed, [Grabar *et al.*, 1999] confirm that a great deal of words in European languages share common Greek and Latin morphemes that allow the generalisation of concepts. As a consequence, it is convenient to define morphological segments as meaningful sequences of characters that should be automatically extracted from corpora. For instance, the following words belong to the same morphological family as they share the same stem [**ocean**] in English: [**ocean***ography*], [**ocean***arium*], [**ocean***ic*] and [**ocean**]. One interesting observation is to see that the same stem can be encountered in Portuguese for the same words presented for English: [**ocean***ografia*], [**ocean***ário*], [**ocean***ico*], [**ocean***o*]. In this context, [**ocean**] would be called a cognate as it is shared by different languages. Thus, cognates or stems may be defined as sequences of characters that co-occur more often than expected by chance.

In order to identify and extract meaningful sequences of words, POS tags, characters and more generally of any textual units, [Dias, 2002] has proposed a statistically-based architecture called SENTA (Software for the Extraction of N-ary Textual Associations) that retrieves, from text corpora, relevant contiguous and non-contiguous textual associations [Dias *et al.*, 2000].

However, the computation of SENTA is hard. Indeed, positional ngrams are ordered sequences of tokens that represent continuous or discontinuous substrings of a corpus computed in a ($2.F+1$)-word size window context. As a consequence, the number of generated substrings rapidly explodes and reaches astronomic figures. For instance, 4.299.742[2] positional ngrams would be generated from a 100.000-word size corpus in a seven-word size window context. It is clear that huge efforts need to be made to process positional ngram statistics in reasonable time and space to tackle real world applications that deal with Gigabytes of data. For that purpose, we propose an implementation that computes positional ngrams statistics in $O(h(F)\ N \log N)$ time complexity and $O(N)$ space complexity. In particular, our architecture is based on the definition of masks that allow virtually representing any positional ngram in the corpus. Thus, we adapt the *Virtual Corpus* approach introduced by [Kit *et al.*, 1998] and apply a suffix-array-like method, coupled to the Multikey Quicksort algorithm [Bentley *et al.*, 1997], to compute positional ngram frequencies. Finally, a multidimensional array is built to easily process both Mutual Expectation and GenLocalMaxs. The evaluation of our C++ implementation has been realized over the CETEMPúblico[3] corpus showing boosted results. For example, it takes 18.23 minutes to compute SENTA for a 1.435.930[4]-word corpus on an Intel Pentium III 900 MHz PC for a seven-word size window context.

2. POSITIONAL NGRAMS

2.1 Principles

In general terms, a positional ngram can be defined as a continuous or discontinuous sequence of tokens[5] in a ($2.F+1$)-token size window context (i.e. F tokens to the left of a pivot token, F tokens to the right of the same pivot token and the pivot token itself). This situation is illustrated in Fig1 for a seven-token size window context.

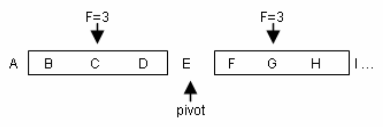


Fig. 1. *7*-token size window context

Thus, any possible substring (continuous or discontinuous) that fits inside the window context and contains the pivot token is a positional ngram. For instance, [E F] is a positional ngram as is the discontinuous sequence [E _ G] where the gap

[2] [Dias, 2002] defines the equation to calculate the number of positional ngrams.

[3] The CETEMPúblico is a 180 million-word corpus of Portuguese.

[4] This represents 61.744.732 positional ngrams.

[5] As token, we mean any textual unit (e.g. character, word, POS tag).

represented by the underline stands for any token occurring between E and G (in this case, F).

2.2 Virtual Representation

The representation of positional ngrams is an essential step towards efficient computation. For that purpose, we propose a reference representation rather than an explicit structure of each positional ngram. The idea is to adapt the suffix representation [Manber *et al.*, 1990] to the positional ngram case. Following the suffix representation, any continuous corpus substring is virtually represented by a single position of the corpus. In fact, the substring is the sequence of tokens that goes from the token referred by the position till the end of the corpus.

Unfortunately, the suffix representation can not directly be extended to the specific case of positional ngrams. One main reason aims at this situation: a positional ngram may represent a discontinuous sequence of tokens. In order to overcome this situation, we propose a representation of positional ngrams based on masks.

One way to represent positional ngrams is to use a set of masks that identify all the valid sequences of tokens in a given window context. Thus, each mask is nothing more than a sequence of 1 and 0 (where 1 stands for a token and 0 for a gap) that represents a specific positional ngram in the window context (Fig2).

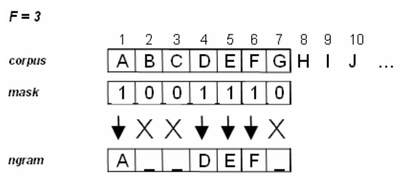


Fig. 2. Masks

As a consequence, any positional ngram can be identified by a position in the corpus and a given mask. Taking into account that a corpus is a set of documents, any positional ngram can be represented by the tuple $\{\{id_{doc}, pos_{doc}\}, id_{mask}\}$ where id_{doc} stands for the document id of the corpus, pos_{doc} for a given position in the document and id_{mask} for a specific mask (Fig3).

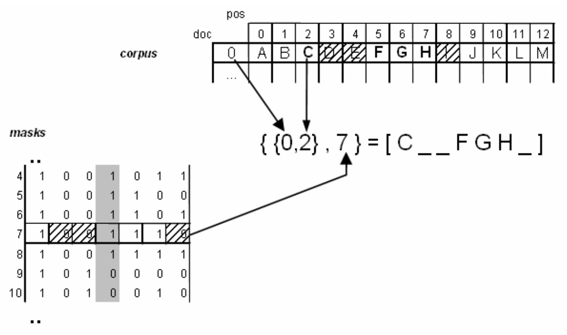


Fig. 3. Virtual Representation

3. COMPUTING FREQUENCY

Counting positional ngrams can be computed following the *Virtual Corpus* Approach [Kit *et al.*, 1998]. A suffix-array structure is sorted using lexicographic ordering for each mask. After sorting, the count of a positional ngram in the corpus is simply the count of adjacent indices that stand for the same sequence (Fig4).

The efficiency of the counting mainly resides in the use of an adapted sort algorithm. For the specific case of positional ngrams, we have chosen to implement the Multikey Quicksort algorithm [Bentley *et al.*, 1997].

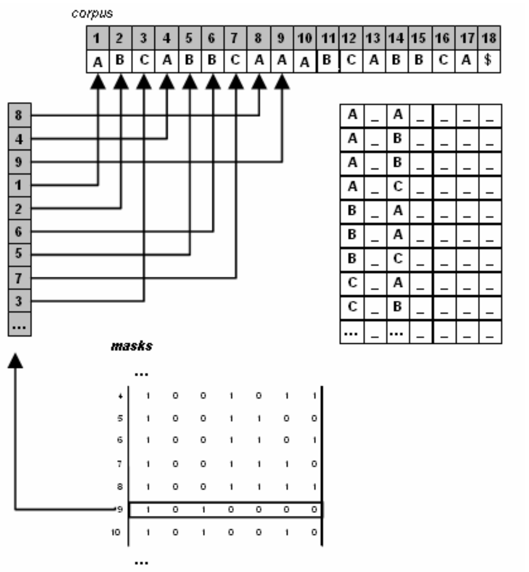


Fig. 4. *Virtual Corpus* for positional ngrams

Different reasons have lead to use the Multikey Quicksort algorithm. First, it performs independently from the vocabulary size. Second, it shows *O(N log N)* time complexity in our specific case. Third, [Anderson *et al.*, 1998] show that it performs better than the MSD radixsort and proves comparable results to the Forward radixsort.

4. COMPUTING THE ACQUISITION PROCESS

The acquisition process is based on an association measure called the Mutual Expectation and an original extraction process called the GenLocalMaxs [Dias 2002]. For that purpose, sub-ngrams (sub-groups of (n-1) tokens) and the super-ngrams (super-groups of (n+1) tokens) of any positional ngram need to be accessed. As a consequence, in order to compute the Mutual Expectation and the GenLocalMaxs, it is necessary to build a data structure that allows efficient search over the space of all positional ngrams. For that purpose, we propose a multidimensional array called **Matrix**. To understand the Matrix itself, we first need to show how the sub-ngrams and the super-ngrams of any positional ngram can be represented. By representing a sub-ngram or a super-ngram, we mean calculating its virtual representation that identifies its related substring.

Representing sub-ngrams: a sub-ngram is obtained by extracting one token at a time from its related positional ngram. Fig5 shows that representing the first three sub-ngrams of the positional ngram {{*0*,*0*},*14*} is straightforward as they all contain the first token of the window context. The only difficulty is to know the mask they are associated to. Knowing this, the first three sub-ngrams would respectively be represented as: {{*0*,*0*},*15*}, {{*0*,*0*},*16*}, {{*0*,*0*},*13*}.
For the last sub-ngram, the situation is different. The first token of the window context is omitted. As a consequence, in order to calculate its virtual representation, we need to know the position of the first token of the substring as well as its corresponding mask.

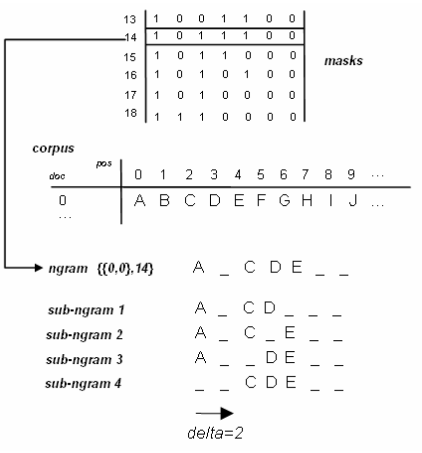
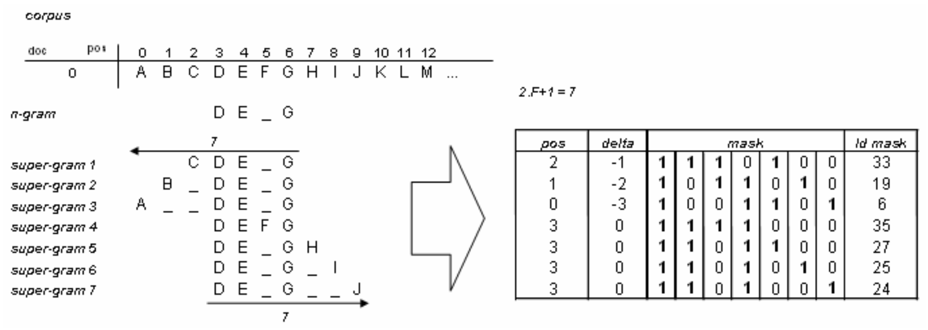


Fig. 5. Sub-ngrams

In this case, the position in the document of the positional sub-ngram is simply the position of its related positional ngram plus the distance that separates the first token of the window context from the first token of the substring. We call *delta* this distance. The obvious representation of the fourth sub-ngram is then {{*0*,*2*},*18*} where the position is calculated as 0+(*delta*=2)=2. So, in order to represent the sub-ngrams of any positional ngram, all we need is to keep track of the masks related to the mask of the positional ngram and the respective *deltas*. Thus, it is clear that for each mask, there exists a set of pairs {id_{mask}, *delta*} that allows identifying all the sub-ngrams of any given positional ngram.

Representing super-ngrams: a super-ngram is obtained by adding a token to its related positional ngram so that it represents a valid positional ngram. Representing super-ngrams is straightforward from the explanation given for sub-ngrams as illustrated in Fig6. Indeed, it is clear that for each mask, there exists a set of pairs {id_{mask}, *delta*} that allows identifying all the super-ngrams of any given positional ngram.



pos	delta	mask							Id mask
2	-1	1	1	1	0	1	0	0	33
1	-2	1	0	1	1	0	1	0	19
0	-3	1	0	0	1	1	0	1	6
3	0	1	1	1	1	0	0	0	35
3	0	1	1	0	1	1	0	0	27
3	0	1	1	0	1	0	1	0	25
3	0	1	1	0	1	0	0	1	24

Fig. 5. Super-ngrams

It is clear that any positional ngram can virtually be represented by a position in the corpus and a given mask. It is also clear that any sub-ngram or super-ngram of a positional ngram can easily be virtually represented given that the pairs {id_{mask}, *delta*} related to its mask are known. Thus, in order to access any positional ngram statistics (*a fortiori* any sub-ngram or super-ngram statistics), an obvious solution is to build a 2-dimension array.

2-dimension array structure: searching for specific positional ngrams in a huge sample space can be overwhelming. To overcome this computation problem, we propose a 2-dimension array (i.e. the Matrix) where lines stand for the masks *ids* and the columns stand for the positions in the corpus. Thus, each cell of the 2-dimension array represents a given positional ngram as shown in Fig6.

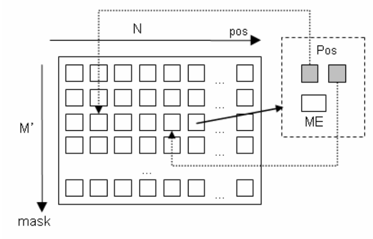


Fig. 6. The Matrix

On one hand, the frequency of each positional ngram can easily be represented by all its positions in the corpus. Indeed, a given positional ngram is a substring that can appear in different positions of the corpus being the count of these positions its frequency. From the previous suffix array-based data structure, calculating all these positions is straightforward. On the other hand, calculating the Mutual Expectation and applying the GenLocalMaxs algorithm is also straightforward and fast as accessing to any positional ngram can be done in $O(1)$ time complexity.

5. EXPERIMENTS

We have conducted a number of experiments of our C++ implementation on the CETEMPúblico Portuguese corpus. The experiments have been realized on an Intel Pentium 900 MHz PC with 390MB of RAM. From the original corpus, we have randomly defined 8 different size sub-corpora that we present in Table 1. For each sub-corpus we have calculated the execution time of different stages of the process for the extraction of word associations: (1) the tokenization that transforms the corpus into a set of integers; (2) the preparation of the mask structure and the construction of the suffix-array data structure; (3) the sorting of the suffix-array data structure and the creation of the Matrix including the calculation of the Mutual Expectation (ME); (4) the execution of the GenLocalMaxs algorithm (Table 2).

6. CONCLUSIONS

In this paper, we have described an efficient implementation to mine relevant patterns from texts based on masks, suffix array-based data structure and multidimensional arrays. Our C++ solution shows that it takes 18.23 minutes to extract word collocations for a 1.435.930-word corpus on an Intel Pentium III 900 MHz for a seven-word size window context. In fact, our architecture evidences *O*(*h(F) N log N*) time complexity and *O(N)* space complexity. It is clear that efficient architectures need to be designed in order to tackle real world applications that need to deal with Gigabytes of text data over the web.

7. ACKNOWLEDGEMENTS

The authors want to thank Professor José Gabriel Pereira Lopes for his valuable advices.

References

[1] G. Dias. 2002. Extraction Automatique d'Associations Lexicales à partir de Corpora. PhD Thesis. New University of Lisbon (Portugal) and University of Orléans (France).

[2] S. Argamon-Engelson, I. Dagan, and Y. Krymolowski. 1998. A Memory-Based Approach to Learning Shallow Natural Language Patterns. In Experimental and Theoretical Artificial Intelligence, 11(3), 67-73.

[3] N. Grabar and P. Zweigenbaum. 1999. Acquisition Automatique de connaissances morphologiques sur le vocabulaire médical. In Traitement Automatique des Langues Naturelles (TALN'99), Institut d'Etudes Scientifiques, Cargèse, France.

[4] G. Dias, S. Guilloré, J.G.P. Lopes. 2000. Mining Textual Associations in Text Corpora. In Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2000), Workshop on Text Mining, August 20-23, Boston, MA, USA, 92-95.

[5] C. Kit and Y. Wilks. 1998. The Virtual Approach to Deriving Ngram Statistics from Large Scale Corpora. In International Conference on Chinese Information Processing Conference, Beijing, China, 223-229.

[6] J. Bentley and R. Sedgewick. 1997. Fast Algorithms for Sorting and Searching Strings. In 8th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orléans.

[7] U. Manber and G. Myers. 1990. Suffix-arrays: A new method for on-line string searches. In First Annual ACM-SIAM Symposium on Discrete Algorithms, 319-327.

[8] A. Anderson and S. Nilsson. 1998. Implementing Radixsort. ACM Journal of Experimental Algorithmics, Vol. 3.

corpus	01	02	03	04	05	06	07	08
Size in Mb	0.2	0.7	2.1	4.2	5.3	6.0	6.7	8.8
# of words	33.095	114.373	342.734	685.274	864.790	978.647	1.092.723	1.435.930
# of ngrams[6]	1.422.827	4.917.781	14.737.304	29.466.524	37.185.712	42.081.563	46.986.831	61.744.732

Table 1. Sub-corpora

corpus	01	02	03	04	05	06	07	08
Tokeniz.	0:00:00	0:00:01	0:00:04	0:00:07	0:00:08	0:00:08	0:00:09	0:00:17
Masks/Suffix	0:00:01	0:00:04	0:00:10	0:00:19	0:00:25	0:00:27	0:00:31	0:00:40
Matrix	0:00:08	0:00:35	0:02:09	0:04:49	0:06:16	0:07:13	0:08:11	0:11:12
GenLocalMaxs	0:00:02	0:00:06	0:00:21	0:00:47	0:01:03	0:01:13	0:01:23	0:06:14
Total	0:00:11	0:00:46	0:02:44	0:06:02	0:07:52	0:09:01	0:10:14	0:18:23

Table 2. Execution Time in (hh:mm:ss)

[6] The window context of the experiment is F=3.