Extracting Multiword's from Large Document Collection based N-Gram

M. Nirmala [1], Dr.E.Ramaraj [2]

[1] *Research Scholar, School of Computer Science and Engineering,*

[2] Associate Professor, School of Computer Science and Engineering,

Alagappa University, Karaikudi, India

***Abstract*—** Multiword terms (MWTs) are relevant strings of words in text collections. Once they are automatically extracted, they may be used by an Information Retrieval system, suggesting its users possible conceptual interesting refinements of their information needs. As a matter of fact, these multiword terms point to relevant information, often corresponding to topics and subtopics in the text collection, and maybe quite useful specially for highly refining generic queries. A new approach is proposed to find collocation from text document. As mentioned earlier, a collocation is just a set of words occurring together more often than by chance in a corpus. Collocations are extracted based on the frequency of the joint occurrence of the words as well as that of the individual occurrences of each of the words in the whole text. Intuitively, when a set of words is extracted as a collocation, then the joint occurrence of the words must be high in comparison to that of the constituent individual words.

***Keywords*—** Multiword terms (MWTs)**,** Information, Collocations, Extraction , Text Document.

I. INTRODUCTION

In Information Retrieval (IR) it is currently accepted that multiword terms enhance IR precision. There are doubts about its role, namely about whether they should work as real indexes or they should play a special role in the refinement phase of user information needs. These multiword terms should constitute a separate document. So, if we want to allow the access to n collections of documents we should produce n documents and each one of these documents should contain the multiword terms of each text collection. Currently used indexing machinery should also index the collection of multiword terms. Acting like this, an information need required from an IR system brings up a number of documents from each collection. Apart from the traditional refinement possibilities using key-words or document descriptors, the system here present may suggest the user the multiword terms containing the words used in the initial query, enabling an information need refinement over the whole collection of collections or over a specific collection. It is left to the user to decide if he/her wants to search over the set of the document collections or over just one collection.

While word based queries show up an enormous number of documents per collection, the use of multiword terms suggested by this IR system dramatically prunes the search space to just a few documents. In this paper we focus on the automatic extraction of multiword terms for any kind of document collection independently of the language used on those documents.

II. RELATED WORK

The Text n-grams extraction is the first part needed for the future use. We are not interested about all n-grams but the specific ones that occur in text at least m-times. It's because we're comparing similarity of documents, respectively the mostly repeated parts of them. In case of huge texts such as 1.5T TREC ClueWeb-B, the use of the ordinary data structures is, such as hash table or search trees, mainly ineffective because the amount of the data cannot be stored in the RAM. Hard drive can be used as a temporary storage where the pre processed data can be stored. The second option is to utilize structures like a B+ tree or Hash table to manage this amount of data. Within the extraction is also mainly stored the information about n-gram position in the document. To save space, it is appropriate to store this information without redundancy. The use of double indexing for this case was shown within data collections protein-10m, protein-100m and protein-1g. Due to the size of the index was reduced 1.9 to 2.7 times and the search speed increased up to 13 times. One opportunity how to process the n-grams is to store complete text of this ngram in a data structure. Effective tool for storing the

data is for example the ternary search tree in which every node stores information about one n-gram character. As shown by tests on collections Google WebIT and English Gigaword corpus is the data structure fast enough.

However, storing whole n-grams in a data structure considerably increases memory requirements. For this case it is better to use two data structures where the words in n-grams are at first converted to unique numbers and only after that the numbers are processed by data structure. The most used data structure to map the words to numbers in n-grams is the hashmap. The Hashmap is, thanks to its properties, fast enough and memory effective to convert words to numbers. It is ideal in cases where there is beforehand known the word count. To store n-grams or the words indexes contained in them is widely used B+ tree. It is no wonder because this data structure was designed to search effectively also with regard to the lack of the memory. In every cell of the B+ tree is stored whole ngram, which is used for comparison during the search process. This attitude was tested on data collection WebIT 5-gram corpus, which contains over 88GB data separated to collection of unigrams to 5-grams. Thanks to word indexing and the use of B+ trees it was managed to store the whole data collection on 598 MB of memory. In this case there is no problem to have the data in memory and thus avoid using slow hard drives.

The creation of the indexes for 5-grams itself takes approximately an hour but it lasts only 2 seconds to look up 1,000 5-grams. One of the key requirements to look up n-grams is the opportunity to use wildcard placeholders, for example when is suitable to look only for particular similarity. When indexing both words and n-grams is first necessary to find a range of words in the first index. However, this is only possible when the indexes are sorted with the words. If this case is fulfilled, it is easy to look up using data structures like B+ tree.

III. EVALUATION METRICS

There have been various evaluation metrics developed and validated for reliability in fields such as MT and summarization (Callison-Burch et al., 2009). While n-gram-based metrics don't capture systematic alternations in key phrases, they do support partial match between key phrase candidates and the reference key phrases. In this section, we first introduce a range of popular n-gram-based evaluation metrics from the MT and automatic summarization literature, which we naively apply to the task of key phrase evaluation. We then present R-precision, an ngram- based evaluation metric developed specifically for key phrase evaluation, and propose a modified version of R-precision which weights n-grams according to their relative position in the key phrase.

3.1 Machine Translation and Summarization Evaluation Metrics

One subtle property of key phrase evaluation is that there is no a priori preference for shorter key phrases over longer key phrases, unlike MT where shorter strings tend to be preferred. Hence, we use the longer NP as reference and the shorter NP as a translation, to avoid the length penalty in most MT metrics. METEOR (Agarwal and Lavie, 2008) is similar to BLEU, in that it measures string-level similarity between the reference and candidate translations. The difference is that it allows for more match flexibility, including stem variation and WordNet synonymy. The basic metric is based on the number of mapped unigrams found between the two strings, the total number of unigrams in the translation, and the total number of unigrams in the reference.

NIST (Martin and Przybocki, 1999) is once again similar to BLEU, but integrates a proportional difference in the co-occurrences for all n-grams while weighting more heavily n-grams that occur less frequently, according to their information value. ROUGE (Lin and Hovy, 2003) — and its variants including ROUGE-N and ROUGE-L—is similarly based on n-gram overlap between the candidate and reference summaries.

For example, ROUGE-N is based on co-occurrence statistics, using higher-order n-grams (n > 1) to estimate the fluency of summaries. ROUGE-L uses longest common subsequence (LCS)-based statistics, based on the assumption that the longer the substring overlaps between the two strings, the greater the similar Saggion et al. (2002). ROUGEW is a weighted LCS-based statistic that prioritizes consecutive LCSes. In this research, we experiment exclusively with the basic ROUGE metric, and unigrams (i.e. ROUGE-1).

3.2 R-precision

R-precision is based on the number of overlapping words between a key phrase and a candidate, as well as the length of each. The metric differentiates three

types of near-misses: Include, Part of and Morph. The first two types are based on an n-gram approach, while the third relies on lexical variation. As we use stemming, in line with the majority of previous work on key phrase extraction evaluation, we focus exclusively on the first two cases, namely include, and part of.

The final score returned by R-precision is:

$$\frac{\text{Number of overlapping word(s)}}{\text{Length of key phrase/candidate}}$$

Where the denominator is the longer of the key phrase and candidate.

3.3 Modified R-precision

R-precision which assigns different weights for component words based on their position in the key phrase (unlike R-precision which assigns the same score for each matching component word). The head noun generally encodes the core semantics of the key phrase, and as a very rough heuristic, the further a word is from the head noun, the less semantic import on the key phrase it has. As such, modified R-precision assigns a score to each component word relative to its position as

$$CW = \frac{1}{N-i+1}$$

Where N is the number of component words in the key phrase and i is the position of the component word in the key phrase (1 = leftmost word).

IV. *EXTRACATION & N-GRAM BASICS*

N-gram Basics:

An N-gram is a character sequence of length n extracted from a document. Typically, n is fixed for a particular corpus of documents and the queries made against that corpus. To generate the N-gram vector for a document, a window n character in length is moved through the text, sliding forward one character at a time. At each position of the window, the sequence of characters in the window is recorded.

For example:

The first four 5-grams in the sentence " character sequences" are " char", "chara", "harac" and "aract". In some schemes, the window may be slid more than one character after each n-gram is recorded. The concept of n-grams was first discussed in 1951 by Shannon. Since then the concept of n-grams have been used in many areas, such as spelling-related applications, string searching, and prediction and speech recognition. Most information retrieval systems are word-based because there are several advantages for word based systems over n-gram based systems.

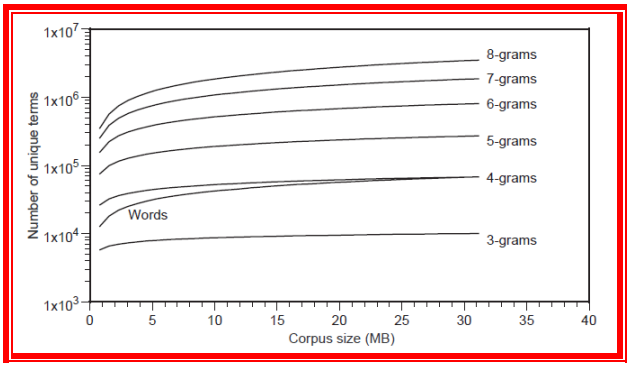

Figure 1: Number of unique terms (words and n-grams) in corpora of varying sizes.

As a result, the index for an n-gram-based system will be much larger than that of a word-based system. Second, stemming techniques can be used in word-based systems but not in n-gram-based systems. Stemming is the process that removes prefixes and suffixes from words in a document or query in the formation of terms in the system's internal model.

This is done to group words that have the same concept meaning, such as "walk", "walked", "walker" and "walking," freeing the user from needing to match the particular form of a word in a query and document. Stemming also reduces the number of unique terms to be indexed. Third, in word-based system, a table can be established for each word to list all of its synonyms. By doing this, if in the query there is a word "home," according to that table, the system will also retrieve the documents containing the word "house." Finally, most word-based systems use stop words. Since stop words appear in most documents, and are thus not helpful for retrieval, these words are usually removed from the internal model of a document or query.

V. *THE LOCAL Maxs ALGORITHM*

The *Local Maxs* Algorithm:

The Local Maxs is an algorithm that works with any text collection as input and automatically

produces multiword terms (MWTs) from that text collection.

In the context of Local Maxs, we define:
An antecedent (in size) of the hole-free n-gram w1, w2...wn, ant((w1...wn)), is a hole-free sub-n-gram of the n-gram w1...wn, having size n-1.
i.e., The (n-1)-gram w1...wn-1 or w2... wn .

A successor (in size) of the hole-free n-gram M = (w1,w2...wn), succ(M), is a hole-free (n+1)-gram N such that M is an ant(N).
i.e., Succ(M) contains the n-gram M and an additional word before (to the left) or after (to the right) M.

Let W be a hole-free n-gram; we say that W is a MWT if4:

$$g(W) \geq g(ant(W)) \wedge g(W) > g(succ(W)) \quad \forall_{ant(W),succ(W)} \quad (\text{if } W\text{'s size} \geq 3)$$

$$g(W) > g(succ(W)) \quad \forall_{succ(W)} \quad (\text{if } W\text{'s size} = 2)$$

Where g(.) is a function that measures the "glue" sticking the words together within the considered n-gram.

The Results

Using the LocalMaxs algorithm and the SCP_f measure, we have attained 84% Precision and 94,039 MWTs from this 2,722,476-word text collection. In this experience the LocalMaxs algorithm was prepared to produce MWTs from 2-grams to 8-grams.

CONCLUSION

In this paper, discussed about the *Local Maxs* algorithm, the SCP measure , it is possible to extract relevant multiword terms. From an Information Retrieval perspective, these multiword terms point to relevant information, often corresponding to topics and subtopics in the text collection. When a set of words is extracted as a collocation, then the joint occurrence of the words must be high in comparison to that of the constituent individual words.This will improve the result and the convergence of the result.

ACKNOWLEDGMENT

First and for most, I own my whole hearted thanks to god for his merciful guidance and abundant blessing.

I am greatly indebted to my parents and department faculties for their great encouragement and co-operation in all aspects to develop this paper.

REFERENCES

1. Efficient in-memory data structures for n-grams indexing . Daniel Robenek, Jan Plato_s, and V_aclav Sn_a_sel, fdaniel.robenek.st, jan.platos, vaclav.snasel.
2. Evaluating N-gram based Evaluation Metrics for Automatic Keyphrase Extraction. Su Nam Kim, Timothy Baldwin, Min-Yen Kan. sunamkim@gmail.com, tb@ldwin.net, kanmy@comp.nus.edu.sg.
3. n-Gram/2L: A Space and Time Efficient Two-Level n-Gram Inverted Index Structure, Min-Soo Kim, Kyu-Young Whang, Jae-Gil Lee, Min-Jae Lee. mskim, kywhang, jglee, mjlee @mozart.kaist.ac.kr.
4. Extracting Multiword Terms from Document collections. Quinta da Torre, 2725, Monte da Caparica, Quinta da Torre, 2725, Monte da Caparica.
5. Automatic Keyword Extraction From Any Text Document Using N-gram Rigid Collocation,Bidyut Das, Subhajit Pal, Suman Kr. Mondal, Dipankar Dalui, Saikat Kumar Shome.International Journal of Soft Computing and Engineering (IJSCE)ISSN: 2231-2307, Volume-3, Issue-2.
6. Advanced Information Extraction with n-gram based LSI,Ahmet Güven, Ö. Özgür Bozkurt, and Oya Kalıpsız.World Academy of Science, Engineering and Technology 17 2008.
7. Evaluating N-gram based Evaluation Metrics for automatic Keyphrase Extraction ,Su Nam Kim, Timothy Baldwin,CSSE University of Melbourne,sunamkim@gmail.com, tb@ldwin.net.Min-Yen Kan School of Computing . National University of Singapore ,kanmy@comp.nus.edu.sg
8. Information Extraction from Web-Scale N-Gram Data ,Niket Tandon. ntandon@mpi-inf.mpg.de ,Gerard de Melo Max Planck Institute for Informatics Saarbrücken, Germany gdemelo@mpi-inf.mpg.de.
9. A Distributed N-Gram Indexing System to Optimizing Persian Information Retrieval ,Mohadese Danesh, Behrouz Minaei, and Omid Kashefi.International Journal of Computer Theory and Engineering, Vol. 5, No. 2, April 2013.
10. Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System,Ethan Miller, Dan Shen, Junli Liu, and Charles Nicholas.University of Maryland Baltimore County,elm,dshen,jliu,nicholas@csee.umbc.edu.