Impact of Ngrams-based indexing on XML retrieval

Mohamed Ben Aouicha[1], Mohamed Tmar[2], and Mohand Boughanem[3]

Institut de Recherche en Informatique de Toulouse,
118 Route de Narbonne, 31062 Toulouse, France
[1] mohamed.benaouicha@irit.fr
[2] mohamed.tmar@isimsf.rnu.tn
[3] bougha@irit.fr

Abstract. We present in this paper a statistical approach of term clustering. This approach is based on a statistical analysis of NGrams shared by a pair of terms and is inspired from the $tf \times idf$ criterion commonly used in information retrieval. Being statistical, the approach is completely independent from the lexical and grammatical characteristics of the language in which documents to be indexed are written. Classical indexing is often based on stemming, which consists of transforming a term into its radical. This allows to provide large issues for customized information access. As for us, we consider that this can be made by building term clusters and perform information retrieval based on this concept. This approach is used for XML retrieval, therefore some experiments have been undertaken into a dataset provided by INEX to show its impact compared to Porter stemming method.

Keywords: NGrams, XML retrieval, INEX'2005, VVCAS.

1 Introduction

Many words have slightly different forms, but their significance remains the same. It is in particular the case of the combined words. For example, the following terms have very similar significance: *implement*, *implementing*, *implementation*. . .

The dissimilarities between these terms is not useful to consider for information retrieval. Contrary, we would like to find documents on *implement* for a information need containing *implementing*. Thus, it is necessary to remove non meaningful parts of a term, i.e. to bring back these words to an identical form called *stem*. It is noticed that these terms have the same stem *implement*. So, if one is able to remove the prefix and suffix from terms, and to keep only the stem, we will have an identical form. It is the idea from which document management systems are using stemmers. There are several ways of stemming terms. The most and commonly used consists on examining only the form of the term, and according to the form, tries to incrementally deduce the stem [2]. The Porter's algorithm removes the suffix from an English term in five steps: the first step consists in transforming plural into singular. The second tries to remove derivations progressively (example: *-ness* that we add behind certain adjectives (happiness), *-able* added behind a verb (adjustable). . .). This algorithm transforms sometimes two different terms into the same form. For

example, *derivate/derive, activate/active*. However, for the majority of cases, the transformation seems reasonable.

We can also use a dictionary while stemming. To show if a sequence of letters at the term end corresponds to an actual suffix, it is enough to attempt to remove or a transform the term in the dictionary. For example, we can accept the rule which replaces *-ation* by *-er* (*transformation*, *elimination*, etc), however, for *vocation*, if this rule is applied, we will obtain *vocer* which does not mean anything in English. We can check in the dictionary if the term exists, if not we do not transform it. This approach is commonly used for the French language [1].

The use of a dictionary involves some advantages, but the majority of information retrieval systems do not have it, and such an electronic dictionary is still not very accessible.

A correct stemming often requires a correct recognition of the grammatical category. Thus, we can think of using an automatic tagger (or a grammatical analyzer) while stemming. One of the possible approaches is to determine the category by appreciating it by probabilities computation. To do so, it is necessary to involve a probabilistic model by using the whole of manually categorized texts (the corpus). This model determines the probabilities that a term belongs to a category according to its form, and the set of terms potentially related to it.

With such category recognition mechanism, we can transform a term into a standard stem instead of simply removing its prefixes, suffixes and infixes.

While indexing, we should transform the terms, select a set of candidate similar terms and measure the similarity between them. The result of indexing is a cluster of terms to which the stem incarnates its centric. This paper aims to define such model and to evaluate its contribution and usability.

The remainder of this paper is organized as follows. Section two presents our approach for term clustering. An XML retrieval system is shown in section three where experiments have been undertaken. Section four provides the experiments and the obtained results. Finally, a summary and future works are given.

2 Lexical equivalence relationship

The main purpose of indexing is to show if a pair of terms is lexically equivalent, thus if their stems are identical. We define an equivalence relationship R in the set of terms of the same language by the following:

$$\forall (t_i, t_j) \in L^2, t_i \, R \, t_j \Leftrightarrow stem(t_i) = stem(t_j) \quad (1)$$

Where $stem(t_.)$ is the stem of a given term $t_.$ and L is a given language (a set of terms).

This relationship is an equivalence (reflexive, symmetrical and transitive). We can then build clusters of terms according to this relationship. If it is considered that the objective of indexing is to build lexical clusters of equivalent terms, it would not be necessary to transform each term into its stem, this transformation has nevertheless the advantage of optimizing the stemming algorithmcomplexity. On the other hand, in order to avoid handling stem identification, it is necessary to define another equivalence relationship which does not con-

sider a term stem. We define a relationship R according to the similarity of each pair of terms by the following:

$$\forall (t_i, t_j) \in L^2, t_i \, R \, t_j \Leftrightarrow sim(t_i, t_j) \geq T_{sim} \quad (2)$$

Where *sim* is related to the terms similarity and T_{sim} is a similarity threshold under which a pair of terms is not potentially similar. It is not possible to affirm that this relation is an equivalence relation because it is neither necessarily symmetrical nor transitive.

2.1 The similarity function

The $tf \times idf$ criterion commonly used in information retrieval reflects the importance degree of a term to a document. A term is important in a document if it often appears in the document (tf) and seldom in the other documents (idf).

This criterion is used in information retrieval to estimate the score of a document to a user query, this reflects the fact that a document is relevant to a user query if they share enough important terms. The $tf \times idf$ criterion is necessary because it provides similarity calculation.

We are inspired from this assumption in order to estimate the similarity between a pair of terms based on the same assumption: two terms are similar if they share enough significant elements.

To adapt this criterion to terms, it is first necessary to define which element characterizes a term as a set of terms characterizes a document. In other words, while a document is a set of terms, a term needs to be expressed by a set of elements. The main problem is to define which represents this element. We chose its NGrams: an NGram is defined by a term factor having length N. For example, the 3Grams of the term *information* are *inf*, *nfo*, *for*, *orm*, *rma*, *mat*, *ati*, *tio* and *ion*. An NGram is significant for a term if it appears at least once in this term and little in the other terms. This implies that the least significant NGrams for a term are those which often appear in the other terms, however the most often met NGams are those that represent the elements to be added to a given term in order to obtain another starting from its stem, for example, the term *formally* is obtained by adding the suffix *-ally* to the stem *form* which appears in a many English terms.

Table 1. $tf \times idf$ used for terms and documents

	Document	Term
Components	Terms	NGrams
tf	frequency of the term in the document	frequency of the NGram in the term
Collection	A set of documents (a corpus)	A set of NGrams (a very large document)

Table 1 presents an analogy of using the $tf \times idf$ criterion in the terms according to its original definition in information retrieval.

The value of N used in our experiments is 2.

2.2 NGram weighting and term similarity

We weight each $N\,Gram_i$ by its $tf \times idf$ in term t_j as follows:

$$p_{ij} = \frac{f_{ij}}{f_i} \tag{3}$$

Where f_i represents the number of distinct terms in the whole collection that contain $N\,Gram_i$ and f_{ij} represents the frequency of $N\,Gram_i$ in term t_j .

The similarity between a pair of terms is given by the following:

$$sim(t_i, t_j) = \sum_{g_k \in t_i \cap t_j} p_{ki} * p_{kj} - \sum_{g_k \in t_i \cap \overline{t_j} \cup \overline{t_i} \cap t_j} p_{ki} * p_{kj} \tag{4}$$

where g_i is an NGram, $t_i \cap t_j$ is the set of NGrams appearing in t_i and t_j and $t_i \cap \overline{t_j} \cup \overline{t_i} \cap t_j$ is the set of NGrams that appear in term t_i and that do not appear in term t_j or vice-versa.

According to equation 4, a term is a set of NGrams, and two terms are equivalent if they share enough strongly weighted NGrams ($\sum_{g_k \in t_i \cap t_j} p_{ki} * p_{kj}$) and if the weight of the non common terms are very low ($\sum_{g_k \in t_i \cap \overline{t_j} \cup \overline{t_i} \cap t_j} p_{ki} * p_{kj}$).

The similarity threshold is a function of the similarity of each term to itself with a multiplicative parameter we define experimentally. The relationship R is then formally defined by the following:

$$\forall (t_i, t_j) \in L^2,\ t_i\ R\ t_j \Leftrightarrow sim(t_i, t_j) \geq S * min\ (sim(t_i, t_i), sim(t_j, t_j)) \tag{5}$$

Where S is a multiplicative parameter less than 1. We typically choose the value allowing as well as possible to bring closer the relationship R in an equivalence relationship: the value allowing to construct clusters with the minimal clustering error.

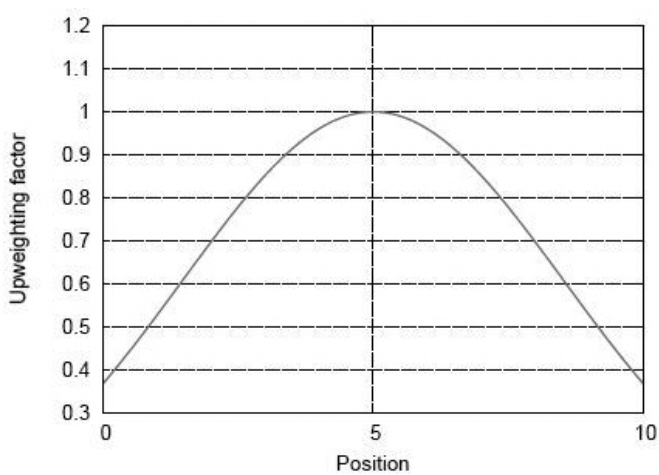

Fig. 1. Evolution of the up weighting factor as function of the relative NGram position.

2.3 NGram weight as function of its relative position

To enhance the NGram position in a given term, we up weight all NGrams that appear close to the middle of the term. In fact, since we consider that prefixes and suffixes are non important for a term, the NGrams that appear near the middle of a term are probably the most representative of its sense. We use the up weighting factor α_{ij} defined as follows:

$$\alpha_{ij} = exp - \left(\frac{p_{ij} - \mu_j}{\sigma_j} \right)$$

where p_{ij} is the position of NGram i in term j, μ_j is the mean average position of an NGram that appears in the term j and σ_j is the standard squared deviation of positions in term j. After some basic computations, μ_j and σ_j are both $\frac{l_j}{2}$ where l_j is the length of term j. As function of the term length, the evolution of the up weighting factor is shown by figure 1 in which we consider a term length 10.

The up weighting factor is taken into account by multiplying the NGram weight by it. In our experiments, we evaluate the NGram weight with and without the up weighting factor. These experiments have been undertaken into an XML retrieval system that we introduce in the following section.

3 XML retrieval model

Extensible Markup Language (XML) [7] is becoming widely used as a standard document format in many application domains. We believe since few years that a great volume of static and dynamic data were produced in XML.

Therefore, XML information retrieval becomes more and more essential [8]. XML documents covers a big part not only on the web, but also on modern digital libraries, business to business and business to consumer software and essentially on Web services oriented software. This is due to the great importance of structured information.

In XML retrieval [9] [10], we take into account the structure and the content of each XML element [3] [4]. Our approach separates between content and structure since indexing queries and documents [6]. Structure retrieval is based on exact matching of flexible representations of a pair of XML trees, which provides the same result as flexible matching of the original representations. Text retrieval however is based on traditional term weighting functions adapted to structured documents. Each XML element text is propagated towards its ascendants. We use the XML tree properties to down weight each propagated term.

4 Experiments and results

Experiments have been undertaken into a dataset provided by INEX. It contains 16819 articles taken from IEEE publications in 24 journals covering the period of 1995-2004 and totaling about 750 megabytes, and 87 queries (40 for CO+S and CO tasks and 47 for CAS task on which we place the emphasis in this paper).

The INEX metric we use for evaluation is based on the normalized extended

cumulated gain (*nxCG*) and the mean average effort precision (*M Aep*). The extended cumulated gain (xCG) metrics are a family of metrics that are an extension of the cumulated gain (CG) based metrics [5] and which aim to consider the dependency of XML elements (e.g. overlap and near-misses) within the evaluation.

Figure 2 shows that the obtained results on MAep are relatively closed to the obtained results by using the Porter's stemming method. This shows that our term clustering method is effective and that the obtained clusters could replace the stem. However, without considering the up weighting parameter α.., the obtained results are slightly better then whose obtained with considering it.

According to the *nxCG*[50], figure 3 shows that the obtained results are better than the obtained results by using the Porter stemming method for all gain-recall values less than 0.3. We notice a clear advantage of using the up weighting factor. For gain-recall values greater than 0.3, a slight advantage to the use of the term up weighting factor is observed.

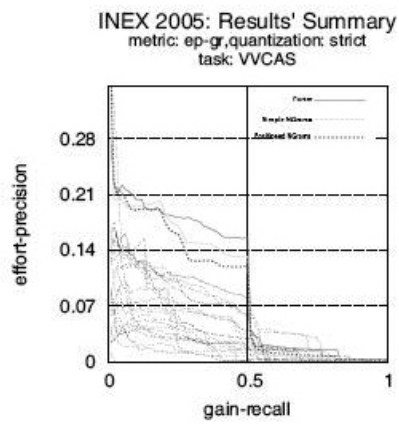

Fig. 2. Comparative MAep based results with official INEX participants

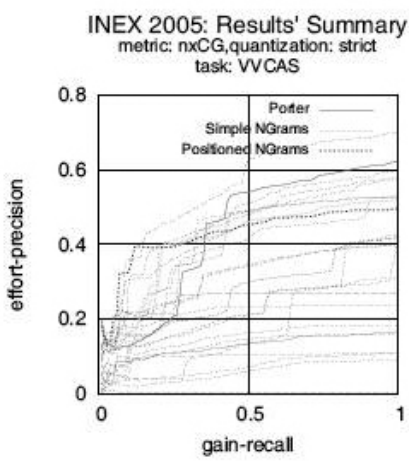

Fig. 3. Comparative nxCG based results with official INEX participants

The experiments were based on unchanging structure scores overall indexing methods (Porter and Ngrams), they do not have any influence on the obtained results. We have observed however a great variation on the content scores which have affected the obtained results.

5 Conclusion

In this paper, we have presented a term clustering method based on estimating the similarity between a given pair of terms. The method is independent from the language, which is not the case of the commonly used methods and especially Porter's method. We have undertaken some experiments on an XML retrieval system and the obtained results show the effectiveness of our approach. The obtained results are closed to those obtained by using Porter's stemming method with slight improvements according to *nxCG*[50]. We have obtained some improvements in taking into account the position of each NGram relatively in the term in which it appears. The main advantage of our study is that our approach could be used for several natural languages. As future work, we plan to improve our approach by using the order of the NGrams in a term. The main assumption of language modelling in information retrieval is to process a query as an ordered list of terms instead of a set of terms. We plan to show the effectiveness of considering a term as an ordered list of NGrams.

References

1. J. Savaoy. Indexation manuelle et automatique : une ´evaluation comparative base sur un corpus en langue franaise Actes 2i`eme COnf´erence en Recherche d'Information et Applications CORIA'05, Grenoble, mars, pages 9-23 (2005).
2. M. F. Porter. an algorithm for suffix stripping Program 14, pages 130-137, (1980).
3. S. Selkow. The tree-to-tree edition problem. Information processing letters, pages 184–186, 1977.
4. J. Wolff, H. Flrke, and A. Cremers. Searching and browsing collections of structural information. Proc. of IEEE advances in digital libraries, Washington, USA, pages 141–150, 2000.
5. Inex - initiative for the evaluation of xml retrieval. http://inex.is.informatik.uniduisburg.de, 2003.
6. R. Luk, H. Leong, T. Dillon, A. Chan, W. Croft, and J. Allan. A survey in indexing and searching xml documents. em Journal of the American Society for Information Science and Technology, 6(53), 2000.
7. World wide web consortium (w3c). extensible markup language (xml) 1.0. http://www.w3.org/TR/REC-xml, 2000.
8. D. Carmel, Y. Maarek, S. Mandelbrod, M. Mass, and A. Soffer. Searching xml documents via xml fragments. Proc. of the 24th annual ACM SIGIR conference on research and development in Information Retrieval, pages 151–158, 2003.
9. Y. Mass, M. Mandelbrod, E. Amitay, D. Carmel, Y. S. Maarek and A. Soffer. JuruXML an XML retrieval system at INEX02. http://inex.is.informatik.uniduisburg.de:2003/proceedings.pdf, pages 73-80, 2003.
10. S. Amer-Yahia, B. Chavdar, J. Dorre and J. Shanmugasundaram. XQuery full-text extensions explained. IBM Systems Journal, pages 335–352, 2006.