Theoretical Computer Science 580 (2015) 75–82

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Finding and enumerating large intersections

Peter Damaschke [1]

Department of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden

A R T I C L E   I N F O

Article history:
Received 15 May 2014
Received in revised form 9 December 2014
Accepted 20 February 2015
Available online 26 February 2015
Communicated by P. Widmayer

Keywords:
Intersection
Disjoint sets
Enumeration
Family of sets
Parameterized algorithm

A B S T R A C T

We study the calculation of the largest pairwise intersections in a given set family. We give combinatorial and algorithmic results both for the worst case and for set families where the frequencies of elements follow a power law, as words in texts typically do. The results can be used in faster preprocessing routines in a simple approach to multi-document summarization.

1. Introduction

This work deals with a natural type of problem: Given a family of n sets called *input sets*, and a number k, compute the k largest pairwise intersections of the input sets. For brevity we call them 2-*intersections*. In particular, in the case $k = \binom{n}{2}$ we want to enumerate all 2-intersections. We will have to state both the assumptions on the set family and the output formats more precisely, see the technical results later on.

The problem has applications in information retrieval, data mining and data compression (see [17] for some pointers), and is also known under the name *top-k set similarity (self-)joins* for the *overlap* similarity measure. (Here we do not consider similarity measures other than overlap, that is, the intersection size.) The fast heuristic algorithms proposed in [17] and similar works rely on the very different element frequencies in real data sets (where the *frequency* of an element is the number of input sets containing it). The first main idea is to process the elements by increasing frequencies f, using the trivial fact that an element appearing in f input sets can appear in at most $\binom{f}{2}$ top 2-intersections, thus the total number of candidate pairs does not grow too much in the beginning. The second main idea called prefix filtering (which is actually a branch-and-bound heuristic) is to bound the maximum possible size of 2-intersections in terms of the number of frequent elements not yet considered, which allows to exclude candidate pairs that provably cannot be among the top k any more.

One question is how many different pairs need to be examined in this way until the k largest 2-intersections are filtered out. Also note that we may only lately detect 2-intersections that solely contain frequent elements, if we only consider the rare elements first. Therefore it is an obvious idea to complement the above incremental heuristic with a separate treatment of the frequent elements. A few elements that appear together in most of the input sets are likely to form already some large 2-intersections which can be precomputed and later combined with the candidate pairs from the low-frequency elements,

E-mail address: ptr@chalmers.se.

[1] Tel.: +46 31 772 5405; fax: +46 31 772 3663.

http://dx.doi.org/10.1016/j.tcs.2015.02.034
0304-3975/© 2015 Elsevier B.V. All rights reserved.

rather than only working with upper bounds based on the number of frequent elements in the single input sets. It fact, it seem to be the frequent elements that mainly determine the final 2-intersection sizes.

This also connects to the so called signature-based heuristics as discussed in [2], however for other similarity measures. The idea is that similar sets must have some common "signatures" on partitions of the input (here we cannot go into any details) and can be detected by that. A combination of these approaches is the "segment bounding algorithm" proposed in [3]. There the set of elements is divided into segments according to frequencies, and then the largest 2-intersections within certain segments are computed by subset hashing, and finally combined following some simple priority rules. The authors report that their algorithm actually runs faster than the algorithm from [17] especially on large data sets.

To conclude, there is empirical evidence that fairly simple heuristics for computing the largest 2-intersections scale very well and are far below the naive quadratic time bound, see also [8,17] and related work cited there. But to our best knowledge no thorough mathematical analysis has been done that would really explain the good running times by, for instance, giving worst-case guarantees under certain structural assumptions valid on real data. Such results would not only provide a better understanding of the heuristics in retrospect. More importantly, they could also suggest algorithms for subproblems that further refine and speed up the practical methods. The present paper aims at some steps in this direction. In particular, the findings in [3] give rise to the algorithmic and combinatorial problems studied here.

Our practical motivation comes from automatic multi-document summarization. An extractive summary of a large collection of sentences, e.g., from news, comments or product reviews, is a small subset of sentences that reflects the most relevant content. Various summarization methods are based on scores, sentence similarity measures, and partly also graph-theoretic concepts, see, for instance, [4,10,15,16,18]. In our own approach[2] we consider sentences, after some preprocessing, as sets of words. Combinations of words appearing in several sentences are important, because several authors wrote, usually independently, about the same issues. Thus we compute the largest intersections of word sets and select some of the involved sentences for a summary, by further simple selection criteria. The approach seems to work reasonably well despite its conceptual simplicity. However, the present paper only deals with related complexity questions rather than the quality of summaries. Specifically, a concern for large text collections is that naive enumeration of all intersections needs quadratic time in the number of sets, whereas we only need the largest ones. The number n of sentences can be some hundreds but also many thousands, and processing times sum up to long waiting times when many collections shall be summarized. Luckily, word frequencies in real texts essentially follow power laws, see [12], and one can take advantage of this property.

In data mining and information retrieval applications, usually the intersection size is normalized by the set sizes, leading to various similarity measures (overlap coefficient, cosine, Dice, Jaccard, and others). However, the absolute intersection size is suitable for our approach to multi-document summarization. Especially in informal text pieces like customer reviews of products it matters which issues have been brought up repeatedly, i.e., which subsets of words appear in several sentences, regardless of what else has been said in these sentences and how long they are. Not the whole sentences need to be similar, but they have to mention the same issues, possibly among others. Nevertheless it could be interesting for further research to adapt the complexity results to other similarity measures. Next, it can be meaningful to weight the words according to, for instance, word type or general importance scores known in advance. In the present paper we study the unweighted case only, however some considerations may be extended more or less straightforwardly to weighted elements. The number k of pairs in the output can be assumed to be small, since for a concise summary we only have use for very few, most significant repeated word combinations, while the others are necessarily neglected.

Overview of results: Section 2 provides some definitions and basic facts. Section 3 deals with the frequent elements. We consider subsets of a fixed set of small size r, such that even times exponential in r are affordable. (In the realm of parameterized complexity, r would be our small parameter.) For computing the k largest 2-intersections of a family of n subsets we derive several worst-case time bounds. It depends on the constellation of r, n, k which of the algorithms is the fastest. We also propose Hasse diagrams as succinct enumerations of the intersecting pairs of sets. Section 4 shows that, if the element frequencies follow a power law, then the number of nonempty pairwise intersections of rare elements, as well as the total number of different pairwise intersections, is $o(n^2)$. We save only a logarithmic factor in the worst case, compared to a $\Theta(n^2)$ result for general set families (Proposition 11). However, the proof uses only the element frequencies and no higher-order structural properties that may yield smaller numbers in real data. Section 5 adds a result in this direction. We consider intersections of size at least 2 and get a time bound where n appears only linearly, multiplied with input parameters that appear to be usually small in text data.

Related topics: A set family naturally corresponds to a bipartite graph whose vertices on both sides represent sets and elements, respectively, and edges represent the element relation. Then, a selection of sets along with the elements in their intersection form a biclique, i.e., complete bipartite subgraph. Thus our problem can be viewed as a pruned biclique enumeration problem, where we only want the bicliques with two vertices on one side but many vertices on the other side. Maximal biclique enumeration in general is well studied [1,7,9,11]. In [6] we have also given an algorithm for the maximal biclique enumeration that runs faster than in the general case under power-law assumptions on the degree sequence. According to an idea in [5] (see also [14]), pairs of similar sets in a set family can be efficiently identified by locality-sensitive hashing. In particular, this applies to set similarity measures where the intersection size is normalized by the size of the sets,

[2] This refers to a project mentioned in the Acknowledgments section.