Finding and enumerating large intersections

Downloaded from: https://research.chalmers.se, 2026-09-06 06:36 UTC

Citation for the original published paper (version of record):
Damaschke, P. (2015). Finding and enumerating large intersections. Theoretical Computer Science, 580: 75-82. http://dx.doi.org/10.1016/j.tcs.2015.02.034

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

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

ARTICLE INFO

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

ABSTRACT

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.

© 2015 Elsevier B.V. All rights reserved.

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.

in different ways. However, sets with just large intersections are not necessarily "similar", as their set differences can still be large. Enumerating intersections is also a subtask in the construction of concept lattices for objects with attributes; we refer to recent work [13] and further references therein. Several output-sensitive algorithms are known for that, however, here we construct only pairwise intersections, and we study the complexity as a function of the number of elements (attributes).

2. Preliminaries

Let R be a fixed *ground set* of r elements. We are given a family of $n \le 2^r$ sets $V \subseteq R$ that we call our *input sets*. A 2-*intersection* is simply an intersection of two input sets. For some prescribed number k we want the k pairs of input sets forming the largest 2-intersections, where ties are broken arbitrarily.

A j-*set* means a set with j elements. A j-*subset* is a set with j elements being a subset of another set understood from context. We define

$$B(r, j) := \sum_{i=0}^{j} \binom{r}{i}.$$

It might be interesting to notice that $B(r, j) = O(2^{H(j/r)})$ for $j \le r/2$ where $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the *binary entropy function*.

In our algorithms we will, to certain sets $V \subseteq R$, assign *records* that store information. We silently assume that a record of a set V is created whenever we want to write something in that record, but it does not yet exist. The records can be stored and retrieved in a dictionary, using the sets V as keys.

At first glance one might think that all 2-intersections are easy to enumerate in $O(r2^r)$ time: For decreasing $j = r, \ldots, 1, 0$ propagate from the records of j-sets V to those of the $j-1$-sets $V \setminus \{v\}$ the information that V is subset of no input set, of exactly one input set (and which), or of at least two input sets, and calculate this information for the $j-1$-sets. However this does not solve the problem. A moment of thinking reveals that in this way we only find maximal 2-intersections, i.e., those not being subsets of other 2-intersections. No matter how many further input sets contain a set V, we still need to figure out whether two of them intersect in exactly V. Thus the problem is more tricky. We will not achieve $O(r2^r)$ time but somewhat larger time bounds depending on various input parameters. In some settings the following fact will be helpful. It is folklore, but for completeness we insert the short and simple proof.

Proposition 1. *For an r-set R, the number of pairs of sets A and B with $A \subseteq B \subseteq R$ is exactly 3^r.*

Proof. We can see this from the Binomial Theorem:

$$\sum_{j=0}^{r} \binom{r}{j} 2^j = \sum_{j=0}^{r} \binom{r}{j} 2^j 1^{r-j} = (2+1)^r = 3^r.$$

Less formally, the assertion also follows from a counting argument: every element of R can, independently, belong to A or $B \setminus A$ or $R \setminus B$. □

The *Hasse diagram* of a partial order, in our case a set family equipped with the $\subset$ relation, is the uniquely determined directed graph representing the partial order relation by directed edges, with transitive edges omitted.

The *frequency* of an element is the number of input sets it belongs to. We say that a set family has a *power law distribution* with exponent $\alpha > 0$ and factor $\tau > 0$, if the following holds true. In the sequence of elements sorted by non-increasing frequencies, the jth element has a frequency at most $\tau n / j^\alpha$. We remark that the exponent α for word frequencies in real text data is usually close to 1.

A *partitioning* divides a set into disjoint nonempty subsets, and the *intersection of two partitionings* consists of all nonempty intersections of the subsets from both.

The 2-*section graph* of a set family has the elements as its vertices, and two vertices are adjacent if and only if the two elements appear together in some input set. Equivalently, every input set gives rise to a clique, and the edge set of the 2-section graph is the union of all edge sets of these cliques. Therefore 2-section graphs are also known as clique graphs, among several other names.

Below we give an overview of the parameters our time bounds will depend on. Some have been already mentioned before, and parameter c will be detailed later in Section 5.

- r: size of the ground set R
- ℓ: maximum size of an input set (clearly $\ell \le r$)
- k: number of largest 2-intersections we want to enumerate
- s: threshold for the size of 2-intersections in the output, that is, the kth largest 2-intersection has s elements
- α, τ: exponent and factor of an element frequency distribution obeying a power law

- f: maximum number of input sets that share one element, also called frequency
- g: maximum number of input sets that share two elements (clearly $g \le f$)
- c: a certain greedy upper bound on the chromatic number of the 2-section graph (where $\ell \le c \le r$ due to the definition of 2-section graph)

3. Enumerating the largest pairwise intersections

A naive enumeration algorithm for the 2-intersections in a family of n input sets being subsets of an r-set R takes all $O(n^2)$ pairs of input sets, computes their intersections in $O(rn^2)$ time and sorts them by size, and it remains to pick the k largest results. Since $n = O(2^r)$, the time can be $O(r4^r)$ in the worst case. However, the worst case for the naive algorithm, $n = 2^r$, actually makes the problem trivial. This indicates that there should exist faster algorithms for "dense" instances with n close to 2^r. Now we could examine the largest potential result sets until k pairs are met. The question is still how much time we need depending on r and k.

Theorem 2. *The k pairs of input sets with the largest 2-intersections can be computed in $O(kr2^r)$ time.*

Proof. First we write every input set V in its own record, that is, in the record of V. This needs $O(rn)$ time in total. Starting from j being the largest size of an input set, we proceed as follows. We copy the contents of the record of each j-set U into the records of all $(j-1)$-sets $U \setminus \{u\}$, $u \in U$. After visiting all records of j-sets we proceed with $j := j-1$. This way, copies of the input sets V are propagated down and written in the records of all subsets $U \subset V$, in the order of decreasing sizes. Next we detail the further steps carried out during this process.

Note that a record may store several input sets. We say that two input sets V and V' *meet* if they appear together in some record. Obviously, they meet for the first time in the record of $V \cap V'$. We maintain in a dictionary all pairs of input sets that have already met. When the record of a set U is visited, we count the number $f(U)$ of input sets stored there. We also form all $\binom{f(U)}{2}$ pairs of them, check which pairs have already met before, and put the new pairs in the dictionary. Obviously the dictionary always holds the largest 2-intersections. Thus we stop as soon as k pairs are collected.

For the time analysis, first note that all write operations into records that store only one input set need together only $O(r2^r)$ time. Once some record U has $\binom{f(U)}{2} \ge k$, we have found the k largest 2-intersections. (This is only a sufficient condition; we may have already collected enough 2-intersections earlier.) Since ties are broken arbitrary, we compute just enough 2-intersections to get a total of k, even if $f(U)$ is larger. Before that moment, or if no such records are encountered, we have $f(U) = O(\sqrt{k})$ in each visited record U. Since at most 2^r records can exist, and all considered sets have at most r elements, this yields the time bound $O(\sqrt{k}r2^r)$ for all set propagations. But we end up with $O(kr2^r)$ time, due to testing in each record which pairs of input sets meet for the first time. □

A minor remark is that some time could be saved as follows. For simplicity we wrote complete input sets in our records. But since we write any input set V only in records of its subsets $U \subseteq V$, it suffices to propagate only the differences $V \setminus U$. While the algorithm itself is straightforward, it provokes some more substantial questions:

Problem 1. Can we compute the k pairs of input sets with the largest 2-intersections faster than in $O(kr2^r)$ time in the worst case?

The problem has two aspects, first of all: Can we improve on the exponential factor 2^r in instances with n being not so close to 2^r? By trivial combinatorics, $\Omega(2^r/\sqrt{r})$ subsets of R have sizes around $r/2$. Now, even in an instance with $k = 1$ and with $2^{r/2}$ input sets of medium size, intuitively it seems difficult to figure out the two input sets having the largest intersection without examining all 2^r pairs of input sets. Although this argument is not conclusive, we conjecture a negative answer.

Furthermore, we may want to improve on the factor k. It came from a worst-case scenario where many records hold $O(\sqrt{k})$ input sets, and we repeatedly detect old pairs of input sets that met before. Can we lower the factor to $\sqrt{k}$ as in the propagation part? An interesting observation is that, for each j, the sizes (larger than j) of the already found 2-intersections together with the numbers $f(U)$ of the j-sets U are sufficient to calculate the number of 2-intersections of size j, with only a $\sqrt{k}$ factor in the time bound. However, the number alone does not tell us what these new 2-intersections are.

In the case that the top 2-intersections are large, the same algorithm has the following refined time bound (the proof is analogous). This case is likely to appear if many subsets of R are input sets.

Corollary 3. *The k pairs of input sets with the largest 2-intersections can be computed in $O(krB(r, r-s))$ time, where s is the size of the kth result.*

A different algorithm is better suited for small s. This time we store the input sets in the records of their subsets with *increasing* sizes.

Theorem 4. *The k pairs of input sets with the largest 2-intersections can be computed in $O(krB(r, s+1))$ time, where s is the size of the kth result.*

Proof. We store the input sets V in the records of their j-subsets $U \subset V$, for increasing j until a stop criterion (see below) is satisfied. Let p_j denote the number of different pairs of input sets that meet in records of j-sets; note that $p_0 \geq p_1 \geq \ldots \geq p_r$, since pairs that meet in a record U also meet in all records of subsets of U. As soon as we detect that $p_j > k$, we leave j and go to $j := j+1$. For every fixed j we proceed in the following order. We take the input sets V one by one, and write V in the records of all its j-subsets $U \subset V$, thereby updating all $f(U)$. Again we always keep $f(U) = O(\sqrt{k})$ for all U, otherwise we know that $p_j > k$.

Eventually we get to the smallest j where $p_j \leq k$. Now we compute the 2-intersections of these p_j pairs, which are clearly all 2-intersections of size at least j, hence the p_j largest ones. In addition we have to compute $k - p_j$ further 2-intersections of size $j-1$; they exist since $p_{j-1} > k$. Note that $s \geq j-1$. Thus we have created at most $B(r, s+1)$ records of j-sets with $j \leq s+1$, and every record stores $O(\sqrt{k})$ input sets. Calculating the p_j costs in total $O(krB(r, s+1))$ time, since $O(kB(r, s+1))$ pairs of input sets, each of size $O(r)$, are examined in the worst case. □

Again it actually suffices to store only the difference $V \setminus U$ rather than V in the record of $U \subset V$, as this provides the same information.

Of course, in general we know s only in hindsight (unless we expect typical intersection sizes from the type of data), however we can just run both algorithms from Theorems 2 and 4 simultaneously and stop when the faster one is ready. Both time bounds can also be written as $O(kr2^{H(s/r)})$, since the binary entropy function H is symmetric on the interval $[0, 1]$.

Suppose that we only want to output the k largest *distinct* 2-intersections, that is, in the sequence S of all 2-intersections sorted by non-decreasing sizes we would take only the distinct ones and omit 2-intersections that equal earlies ones but come from different pairs of sets. We can enumerate them within similar time bounds, with only an extra $O(r)$ factor, due to the following observation: Each 2-intersections U in S involves at most $r+1$ new input sets. This is because any two new input sets V and V' must be disjoint outside U (that is, $(V \cap V') \setminus U = \emptyset$), otherwise they would have appeared earlier in S. Thus, at most $(r+1)k$ input sets form the k largest distinct 2-intersections.

The $O(kr2^r)$ bound in Theorem 2 is better than the naive bounds $O(n^2 r)$ and $O(r4^r)$, as long as $k < n^2/2^r$ and $k < 2^r$, respectively. Next we look at larger k and ask whether we can *enumerate all distinct 2-intersections* faster than in $O(r4^r)$ time in the worst case. As argued in Section 2, a seemingly simple $O(r2^r)$ time algorithm is incorrect. Instead we accomplish the goal to be faster than $O(r4^r)$ by reducing the problem to many instances of the recognition of disjoint input sets.

Lemma 5. *We can find two disjoint input sets, or recognize that no two disjoint input sets exist, in $O(r2^r)$ time.*

Proof. For any ordered pair of sets X, Y, we have $X \cap Y = \emptyset$ if and only if $X \subseteq R \setminus Y$. We write every input set, and also the complement of every input set, in its own record, in $O(r2^r)$ time in total. We also mark whether a record stores an input set or a complement. Starting with $j := r$ we copy the contents of each record of a j-set U, that stores the complement of an input set, into the records of all $(j-1)$-sets $U \setminus \{u\}$, $u \in U$, and always decrement j after visiting all j-sets. When two sets meet in a record, we keep only one of them, arbitrarily selected. Clearly, there exists a disjoint pair if and only if some complement of an input set gets into the record of an input set. The $O(r2^r)$ time bound is obvious, too. □

Theorem 6. *All distinct 2-intersections can be enumerated in $O(r3^r)$ time.*

Proof. We write every input set V in the records of all sets $U \subseteq V$. The total number of copies of input sets is bounded by 3^r, due to Proposition 1. Any set U is a 2-intersection if and only if two input sets $V, V' \supseteq U$ exist with $(V \setminus U) \cap (V' \setminus U) = \emptyset$. After ignoring the elements of U themselves in the record of U, by Lemma 5 this condition can be checked in $O(j2^j)$ time, where $j = r - |U|$. Hence the same calculation as in Proposition 1 again yields the time bound. □

Coming back to the problem of computing the k largest 2-intersections for prescribed k, we observe that Theorem 6 beats Theorem 2 for $k > 1.5^r$, however, the method in Theorem 6 outputs only the 2-intersections themselves, but not all pairs of input sets that yield these 2-intersections (except one representative pair for each). This would not even be desirable. An explicit list of all pairs would be too large since, trivially, it can have size $O(4^r)$ again. This motivates the computation of the Hasse diagram of all input sets and their 2-intersections. Note that the Hasse diagram provides a succinct enumeration of pairs in the following sense: $U = V \cap V'$ holds if and only if U is the highest common descendant of V and V'. Thus the structure yields more information, but its computation is not more expensive; we will preserve the time bound from Theorem 6.

Proposition 7. *The Hasse diagram of input sets in R can be constructed in $O(3^r)$ time.*

Proof. For every input set V we do the following separately. We write a bit 1 in the records of all subsets of V, in the order of decreasing size. Whenever a bit 1 enters the record of another input set, we change the bit to 0 and continue

propagating it to all subsets. It is straightforward to see that $U \subset V$ is connected to the fixed V by an edge in the Hasse diagram if and only if U receives a 1 from some superset, but no 0. The total number of bits emanating from all input sets V is at most 3^r by Proposition 1. □

Theorem 8. *The Hasse diagram of a family of input sets in R and all their 2-intersections can be constructed in $O(r3^r)$ time.*

Proof. Use Theorem 6 to get all 2-intersections, and then compute the Hasse diagram, by applying Proposition 7 to the (original) input sets and their 2-intersections. □

An obvious question is, again, whether the worst-case time bounds can be improved. First we can prove that Lemma 5 is optimal.

Proposition 9. *Any algorithm that finds two disjoint input sets needs $\Omega(2^r)$ time, with polynomial factors neglected.*

Proof. Consider instances where r is even, and all input sets have exactly $r/2$ elements. Then, for every input set there is only one possible other input set that might be disjoint to it. Trivially, if at most half of the possible sets of size $r/2$ are input sets, we can identify such a pair (or report that none exists), only by looping through all input sets. By simple combinatorics, the number of sets of cardinality $r/2$ is still $\Omega(2^r)$ up to polynomial factors. □

However, from Proposition 9 we cannot conclude optimality of the base 3 in Theorem 6. While deciding whether any fixed set U is a 2-intersection of input sets V is equivalent to the disjointness problem for the $V \setminus U$ with $V \supseteq U$ (see the proof of Theorem 6) for which we know the optimal base, the problem instances for all sets U together are not "independent", and there might exist a different algorithm not using Lemma 5. Therefore we raise:

Problem 2. Can we enumerate all distinct 2-intersections faster than in $O(r3^r)$ time in the worst case?

At the end of this section we mention the same type of problem with the maximum input set size ℓ as the parameter. We proceed as in Theorem 2. The only difference is in the number of records as a function of ℓ, where we incur a factor n. The resulting bound still beats the trivial $O(n^2\ell)$ when ℓ is logarithmic in n.

Theorem 10. *Given n input sets of size at most ℓ (but with an arbitrarily large union), the k pairs of input sets with the largest 2-intersections can be computed in $O(nkr2^\ell)$ time.*

4. Number of pairwise intersections under a power law

In this section we show that set families where element frequencies follow a power law (recall the motivation from text data in Section 1) exhibit considerably fewer than $\binom{n}{2}$ different 2-intersections. Before that, we observe that set families in general can have the maximum possible number of 2-intersections, already on small ground sets of logarithmic size. This is not at all surprising but worth mentioning, as it contrasts to the result for power laws.

Proposition 11. *For any $r > 7.23 \log_2 n$ there exist families of n input sets of an r-set R where all $\binom{n}{2}$ possible 2-intersections are different.*

Proof. This is a simple case of the Probabilistic Method. Create n sets by deciding for each element of R whether it belongs to a set or not, independently and with probability 1/2. Note that some of the n random sets may happen to be identical, but this will not affect the argument. Consider any four sets A, B, C, D. The event $A \cap B = C \cap D$ appears with probability $(5/8)^r$; just count the possible cases for each element of R. Similarly, consider any three sets A, B, C. The event $A \cap B = B \cap C$ appears with probability $(3/4)^r$. We have less than n^4 quadruples of sets. By the union bound, the probability that some of them has equal 2-intersections is at most $n^4(5/8)^r$. Similarly, the probability that some of the triples of sets has equal 2-intersections is at most $n^3(3/4)^r$. Hence, if $n^4(5/8)^r + n^3(3/4)^r < 1$ then all 2-intersections are distinct with some positive probability, which implies the existence of a set family as claimed. Moreover, the n random sets are distinct in this case. In particular, it is sufficient that: $n^4(5/8)^r < 1/2$ and $n^3(3/4)^r < 1/2$. This is equivalent to $4\log_2 n - r\log_2(8/5) < -1$ and $3\log_2 n - r\log_2(4/3)^r < -1$. Numerical calculation shows that the former inequality implies the latter one, and it also yields the factor at $\log_2 n$. □

Theorem 12. *For any fixed $\alpha > 1/2$ and $\tau > 0$ we have: Any set family where the frequencies follow a power law distribution with exponent α and factor τ has $O(n^2/(\log n)^{2\alpha-1})$ different 2-intersections, moreover, they can be enumerated in $O(n^2/(\log n)^{2\alpha-1})$ time.*

Proof. For some fixed integer r to be discussed later, we call the r elements with the highest frequencies (ties are broken arbitrarily) the *frequent elements*, all others are called the *rare elements*. Consider the set family induced by the set R of frequent elements, that is, by ignoring the rare elements. Note that the resulting set family on R may contain sets multiple times. A *module* is any such sub-family of input sets having exactly the same elements in R. Clearly, at most 2^r modules exist, and they produce at most 2^r distinct 2-intersections within R. (A trivial side remark is that, if some $V \subseteq R$ appears several times, then V is also the 2-intersection of any two copies of itself.)

Next we restrict the input sets to the rare elements instead, that is, we temporarily ignore the frequent elements. We bound the number q of pairs of sets having non-empty 2-intersections (which are not necessarily distinct) of rare elements. Let v be any rare element, with frequency $f = f(v)$. Then, obviously, v is contained in $\binom{f}{2}$ such pairs. Thus we simply have $q \le \sum_v \binom{f(v)}{2}$, where the sum is taken over all rare elements v. In summary, at most 2^r different 2-intersections of frequent elements exist, and at most q pairs of sets may also have rare elements in their intersections. It follows that at most $2^r + q$ different 2-intersections exist.

For any fixed $\alpha > 1/2$, the sum $\sum_{j=1}^{\infty} 1/j^{2\alpha}$ converges; specifically:

$$q < \sum_{j=r+1}^{\infty} \binom{\tau n/j^\alpha}{2} < \frac{\tau^2 n^2}{2} \sum_{j=r+1}^{\infty} 1/j^{2\alpha}$$
$$< \frac{\tau^2 n^2}{2} \int_r^{\infty} \frac{dx}{x^{2\alpha}} = \frac{\tau^2}{(4\alpha-2)r^{2\alpha-1}} \cdot n^2.$$

Since α and τ are fixed, we see that q/n^2 depends on r only, and becomes arbitrarily small as r grows. Since we have at most $2^r + q$ different 2-intersections, we choose $r = \rho \log_2 n$ for some factor $\rho < 2$ and obtain

$$2^r + q < n^\rho + \frac{\tau^2}{(4\alpha-2)\rho^{2\alpha-1}} \cdot \frac{n^2}{(\log_2 n)^{2\alpha-1}}.$$

This yields the first assertion. As for the time bound, recall that $q \le \sum_v \binom{f(v)}{2}$, the total number of occurrences of rare elements in pairs of sets. We can simply collect them in $O(q)$ time, to obtain the 2-intersections restricted to the rare elements. It remains to pair them up with the 2-intersections of the frequent elements, solely determined by the modules the affected input sets belong to. (We omit some straightforward details.) For simplicity we may use trivial enumeration of them in $O(r4^r)$ time and choose some $\rho < 1$, such that the term $4^r = 4^{\rho \log_2 n} = n^{2\rho}$ remains below the desired bound. □

Consider input sets of roughly equal sizes ℓ. (This is not a severe restriction in the summarization application, where the sentences have a typical maximum length ℓ, and shorter sentences can be padded.) Compared to a naive $O(\ell n^2)$ time enumeration of the 2-intersections we save some $\Theta(\ell(\log n)^{2\alpha-1})$ factor, which makes already a difference. Note that ℓ does not appear in the time bound in Theorem 12.

5. Enumerating pairwise intersections through a coloring

In this section we consider set families where the n input sets have maximum size ℓ, but the ground set can be larger.

We will use a coloring of the 2-section graph, not necessarily an optimal one. (Graph coloring is an NP-hard problem.) Instead we take a greedy coloring obtained as follows. Index the colors by integers $1, 2, 3, \ldots$, sort the elements by increasing frequencies (in $O(\ell n)$ time using Bucketsort), and give every element the first available color. Let c denote the number of colors used. The greedy coloring can be obtained as follows. Imagine a table whose rows and columns correspond to the input sets and colors, respectively. Initially it is empty. For every element v we search, in the rows of all input sets $V \ni v$, for the first column where all places in these rows are still free, and we fill these entries with the symbol v. There exist $O(\ell n)$ pairs (v, V) of elements v and input sets $V \ni v$ and c colors, hence this needs only $O(c\ell n)$ time.

Trivially, we have $c \ge \ell$. But in set families of the rare words in texts we can expect that the greedy chromatic number c does not exceed ℓ much. Roughly speaking, rare words are spread over the different texts in a quasi random fashion, such that the table will be filled up quite well from left to right by the greedy procedure.

Besides c we use a second input parameter of the set family that appears to be small in text data: Recall that g denotes the maximum number of input sets that share two elements. Equivalently, this is the maximum frequency of pairs of elements, or the largest 2-intersection in the dual set family. Clearly, g is bounded by the maximum frequency f, but can be much smaller.

Theorem 13. *Given a c-coloring of the 2-section graph of family of input sets, the number of 2-intersections of size at least 2 is $O(gc^2 n)$, and we can enumerate them in $O(gc^2 n)$ time.*

Proof. Let I denote the set of rows in the aforementioned table, thus we have one row for each input set, and $|I| = n$. In any fixed column (color), the sets of entries containing the same symbol v, plus the set of entries remaining empty, naturally

form a partitioning of I. Now we take each of the $O(c^2)$ pairs of colors and compute in $O(n)$ time the intersection π of their two partitionings; details are straightforward. The sets in π are exactly the row sets of those input sets containing any specific pair of elements, of the two considered colors. Trivially, the sum of sizes of all these row sets is $O(c^2 n)$. Since each of them comprises at most g rows, the sum of squares of their sizes is $O(gc^2 n)$, as can be seen by convexity of the square function or by a simple counting argument. In other words, we have found all quadruples (u, v, U, V) with $u, v \in U \cap V$, and their number is $O(gc^2 n)$. In order to construct the 2-intersections, it remains to put together these quadruples in an obvious way. □

If c remains close to ℓ (see above), the time bound becomes $O(g\ell^2 n)$, as opposed to the naive $O(\ell n^2)$ bound.

Acknowledgements

This work has been supported by the Swedish Foundation for Strategic Research (SSF) through Grant IIS11-0089 for a data mining project entitled “Data-driven secure business intelligence”. I would like to thank our Algorithms group and collaborators at the companies Recorded Future and Findwise for discussions, Olof Mogren and Azam Sheikh Muhammad for their extensive work on text summarization, the master’s students Emma Bogren and Johan Toft for their literature investigations and implementations of their new heuristics, and the anonymous referees for valuable comments.

References

[1] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P.L. Hammer, B. Simeone, Consensus algorithms for the generation of all maximal bicliques, Discrete Appl. Math. 145 (2004) 11–21.
[2] A. Arasu, V. Ganti, R. Kaushik, Efficient exact set-similarity joins, in: U. Dayal, et al. (Eds.), 32nd Int. Conf. on Very Large Data Bases, VLDB 2006, ACM, 2006, pp. 918–929.
[3] E. Bogren, J. Toft, Finding top-k similar document pairs – speeding up a multi-document summarization approach, Master’s thesis, Dept. of Computer Science and Engin., Chalmers, Göteborg, 2014.
[4] M. Bonzanini, M. Martinez-Alvarez, T. Roelleke, Extractive summarisation via sentence removal: condensing relevant sentences into a short summary, in: G.J.F. Jones, et al. (Eds.), 36th Int. ACM SIGIR Conf. on Research and Development in Info Retrieval, SIGIR 2013, ACM, 2013, pp. 893–896.
[5] A.Z. Broder, On the resemblance and containment of documents, in: Compression and Complexity of Sequences, SEQUENCES’97, IEEE Comp. Society, 1997, pp. 21–29.
[6] P. Damaschke, Enumerating maximal bicliques in bipartite graphs with favorable degree sequences, Inform. Process. Lett. 114 (2014) 317–321.
[7] V.M.F. Dias, C.M.H. de Figueiredo, J.L. Szwarcfiter, On the generation of bicliques of a graph, Discrete Appl. Math. 155 (2007) 1826–1832.
[8] T. Elsayed, J. Lin, D.W. Oard, Pairwise document similarity in large collections with MapReduce, in: 46th Annual Meeting Assoc. Comp. Ling., Short papers, ACL 2008, Assoc. Comp. Ling., 2008, pp. 265–268.
[9] A. Gely, L. Nourine, B. Sadi, Enumeration aspects of maximal cliques and bicliques, Discrete Appl. Math. 157 (2009) 1447–1459.
[10] D. Ji, Y. Nie, Sentence ordering based on cluster adjacency in multi-document summarization, in: 3rd Int. Joint Conf. Nat. Lang. Proc., IJCNLP 2008, Assoc. Comp. Ling., 2008, pp. 745–750.
[11] J. Li, G. Liu, H. Li, L. Wong, Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: a one-to-one correspondence and mining algorithms, IEEE Trans. Knowl. Data Eng. 19 (2007) 1625–1637.
[12] W. Li, Random texts exhibit Zipf’s-law-like word frequency distribution, IEEE Trans. Inform. Theory 38 (1992) 1842–1845.
[13] J. Muangprathub, A novel algorithm for building concept lattice, Appl. Math. Sci. 8 (2014) 507–515.
[14] A. Rajaraman, J. Ullman, Mining of Massive Datasets, Cambridge Univ. Press, 2011.
[15] X. Wan, J. Yang, Multi-document summarization using cluster-based link analysis, in: S.H. Myaeng, et al. (Eds.), 31st Int. ACM SIGIR Conf. on Research and Development in Info Retrieval, SIGIR 2008, ACM, 2008, pp. 299–306.
[16] D. Wang, S. Zhu, T. Li, SumView: a web-based engine for summarizing product reviews and customer opinions, Expert Systems With Applications 40 (2013) 27–33.
[17] C. Xiao, W. Wang, X. Lin, H. Shang, Top-k set similarity joins, in: Y.E. Ioannidis, D.L. Lee, R.T. Ng (Eds.), 25th Int. Conf. on Data Engin., ICDE 2009, IEEE, 2009, pp. 916–927.
[18] W. Yih, J. Goodman, L. Vanderwende, H. Suzuki, Multi-document summarization by maximizing informative content-words, in: M.M. Veloso (Ed.), 20th Int. Joint Conf. on Artif. Intell., IJCAI, 2007, pp. 1776–1782.