Theoretical Computer Science 648 (2016) 8–25

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient optimally lazy algorithms for minimal-interval semantics ☆

Paolo Boldi, Sebastiano Vigna *

Dipartimento di Informatica, Università degli Studi di Milano, Italy

ARTICLE INFO

Article history:
Received 29 May 2015
Received in revised form 3 May 2016
Accepted 29 July 2016
Available online 9 August 2016
Communicated by P. Widmayer

Keywords:
Lazy algorithms
Information retrieval
Lattices

ABSTRACT

Minimal-interval semantics [8] associates with each query over a document a set of intervals, called *witnesses*, that are incomparable with respect to inclusion (i.e., they form an antichain): witnesses define the minimal regions of the document satisfying the query. Minimal-interval semantics makes it easy to define and compute several sophisticated proximity operators, provides snippets for user presentation, and can be used to rank documents. In this paper we provide algorithms for computing conjunction and disjunction that are linear in the number of intervals and logarithmic in the number of operands; for additional operators, such as ordered conjunction and Brouwerian difference, we provide linear algorithms. In all cases, space is linear in the number of operands. More importantly, we define a formal property of *optimal laziness*, and either prove it, or prove its impossibility, for each algorithm. We cast our results in a general framework of finite antichains of intervals on total orders, making our algorithms directly applicable to other domains.

1. Introduction

Modern information-retrieval systems, such as web search engines, rely on *query expansion*, an automatic or semi-automatic mechanism that aims at rewriting the user intent (i.e., a set of keywords, maybe with additional context such as geographical location, past search history, etc.) as a structured query built upon a number of operators. The simplest case is that of the *Boolean model*, in which operators are just conjunction, disjunction and negation: as an example, the set of keywords provided by the user might be expanded as disjunctions of syntactically or semantically similar terms, and finally such disjunctive queries would be connected using a conjunction operator. The semantics provided by this model is simply either the value "true" or the value "false" (in the example, "true" is returned if the document contains at least one term from each disjunction).

When a document satisfies a query, however, the Boolean model fails to explain *why* and *where* the query is satisfied: this information is compressed in a single truth value. *Minimal-interval semantics* is a richer semantic model that uses *antichains*[1] *of intervals of natural numbers* to represent the semantics of a query; this is the natural framework in which operators such

☆ A preliminary version of some of the results in this paper appeared in [5]. All algorithms have been significantly simplified, getting rid of the double queues of [5]. The notion of optimal laziness and all related results are entirely new. The algorithm for the AND operator has been improved during the proofs of optimality (it was not optimally lazy in the formulation of [5]).

* Corresponding author.
E-mail address: sebastiano.vigna@unimi.it (S. Vigna).

[1] An *antichain* of a partial order is a set of elements that are pairwise incomparable.

http://dx.doi.org/10.1016/j.tcs.2016.07.036
0304-3975/

as ordered conjunction, proximity restriction, etc., can be defined and combined freely. Each interval is a *witness* of the satisfiability of the query, and defines a region of the document that satisfies the query (positions in the document are numbered starting from 0, so regions of text are identified with sets of consecutive integers, a.k.a. intervals).

Consider, for example, the document given by the start of the well-known rhyme

Pease porridge hot! Pease porridge cold!

If we query this document with the keyword "*hot*", we just get the Boolean answer "true". But a more precise answer would be "*hot* appears in the third position of the document": formally, this is described by the interval $[2\,..\,2]$. Sometimes, a query will be satisfied in multiple parts of the document; for example, the query "*pease*" has answer $\{[0\,..\,0], [3\,..\,3]\}$. If we consider *two* keywords things become more interesting: when we submit the conjunctive query "*pease* AND *porridge*" the answer will be $A = \{[0\,..\,1], [1\,..\,3], [3\,..\,4]\}$. Of course, there are more intervals containing both *pease* and *porridge*, but they are omitted because they contain (and thus they are less informative than) one of the intervals in A. The latter observation leads us to considering sets of intervals that are incomparable with respect to inclusion, that is, antichains with respect to $\subseteq$.

This approach has been defined and studied to its full extent by Clarke, Cormack and Burkowski in their seminal paper [8]. They showed that antichains have a natural lattice structure that can be used to interpret conjunctions and disjunctions in queries. Moreover, it is possible to define several additional operators (proximity, followed-by, and so on) directly on the antichains. The authors have also described families of successful ranking schemes based on the number and length of the intervals involved [7].

The main feature of minimal-interval semantics is that, by its very definition, an antichain of intervals cannot contain more than w intervals, where w is the number of words in the document. Thus, it is in principle possible to compute all minimal-interval operators in time linear in the document size. This is not true, for instance, if we consider different interval-semantics approaches in which *all* intervals are retained and indexed (e.g., the PAT system [10] or the `sgrep` tool [13]), as the overall number of output regions is quadratic in the document size.

In this paper, we attack the problem of providing efficient lazy algorithms for the computation of a number of operators on antichains. As a subproblem, we can compute the proximity of a set of terms, and indeed we are partly inspired by previous work on proximity [19,18]. Our algorithms are linear in the number of input intervals. For conjunction and disjunction, there is also a multiplicative logarithmic factor in the number of input antichains, which however can be shown to be essentially unavoidable in the disjunctive case. The space used by all algorithms is linear in the number of input antichains (in fact, we need to store just one interval per antichain), so they are a very extreme case of stream transformation algorithms [2,12]. Moreover, our algorithms satisfy some stringent formal *laziness* properties.

Note that from a practical viewpoint laziness makes our algorithms very attractive when paired with an index structure (see, e.g., quasi-succinct indices [21]) that provides lazy I/O for reading positions. For example, it is possible to decide that a set of terms appear within a certain proximity bound *without* reading all positions: if the underlying index is lazy, our algorithms limit the I/O as much as possible. In the open-source world, the semantic engine Mímir [20] is based on MG4J [4], which contains our implementation of such algorithms.

In Section 2 we briefly introduce minimal-interval semantics, and provide some examples and motivations. The presentation is rather algebraic, and uses standard terms from mathematics and order theory (e.g., "interval" instead of "extent" as in [8]). The resulting structure is essentially identical to that described in the original paper [8], but our systematic approach makes good use of well-known results from order theory, making the introduction self-contained. For some mathematical background, see, for instance, [3,9].

Another advantage of our approach is that by representing abstractly regions of text as intervals of natural numbers we can easily highlight connections with other areas of computer science: for example, antichains of intervals have been used for testing distributed computations [14]. The problem of computing operators on antichains has thus an intrinsic interest that goes beyond information retrieval. This is the reason why we cast all our results in the general framework of antichains of intervals on arbitrary (totally) ordered sets.

Finally, we present our algorithms. First we discuss algorithms based on queues, and then greedy algorithms.[2]

2. Minimal-interval semantics

Given a totally ordered set O, let us denote with $\mathscr{C}_O$ the set of intervals[3] of O that are either empty or of the form $[\ell\,..\,r] = \{x \in O \mid \ell \le x \le r\}$. Our working example will always be $O = W$, where $W = \{0, 1, \ldots, w-1\}$ and w represents the number of words in a document, numbered starting from 0 (see Fig. 1); elements of $\mathscr{C}_W$ can be thought of as regions of text.

Intervals are ordered by containment: when we want to order them by *reverse* containment instead, we shall write $\mathscr{C}_O^{\mathrm{op}}$ ("op" stands for "opposite"). Given intervals I and J, the interval *spanned* by I and J is the least interval containing I and J (in fact, their least upper bound in $\mathscr{C}_O$).

[2] A free implementation of all algorithms described in this paper is available as a part of MG4J [4] (http://mg4j.di.unimi.it/) and LaMa4J (http://lama4j.di.unimi.it/).

[3] A subset X of O is an *interval* if $x, y \in X$ and $x \le z \le y$ imply $z \in X$.