Without the context definition, however, hash join is more frequent, because hash join is an important problem in the database history. As a result, MapReduce becomes more discriminative. By TDC, the changes in term discrimination will evidently result in different rankings.

Our ranking framework radically changes query processing and poses new challenges on query efficiency. In conventional query processing, a canonical optimization is to evaluate the most selective sub-expression first, so that the number of intermediate results is dramatically reduced. A query optimizer may interleave text predicates and XML query constructs in a query plan to achieve optimality. This strategy, however, is not valid in our ranking framework. Unlike prior ranking schemes which collect keyword statistics at indexing time, our ranking framework treats a fragment satisfying the pattern as a “document”, and therefore has to collect pattern-specific statistics. This means that the tree pattern must be evaluated holistically, even if some of the text predicates are selective. This restriction can lead to efficiency issues, when the tree pattern is not selective and the computation of ranking scores need to materialize and access a large number of “documents” to collect statistics.

In this paper, we exploit view-based techniques to overcome the challenges of query efficiency. At the core of the challenges is the computation of pattern-specific statistics, some of which (e.g., a variant of document frequency) demand aggregations over all satisfied XML fragments! This is analogous to analytical queries in RDBMSs, which often need to aggregate parameters over a large number of tuples satisfying predicates. Inheriting the OLAP idea, we study the problem of using aggregation views (aka 'bases') to compute pattern-specific statistics. We show that due to XML query semantics and the properties of the computations, data cubes have limited capability in our setting. Then we study using non-aggregation views for query evaluation, and propose a columnar view storage. This aligns with the recent emergence of relational column stores, which have demonstrated tremendous potential for analytical queries. The proposed storage inherits major benefits of relational column stores, meanwhile maintaining special structures to support XPath semantics.

Main contributions of this paper are:

1. We propose a new ranking framework that is consistent with IR heuristics for tree pattern