Progetto 12

Your company is involved in the design of a new search engine. Your search engine features both classic searches and sponsored searches.

For the classic search, the company needs to implement both a matching algorithm and a ranking algorithm. For the matching algorithm, your company has to choose between two alternatives:

- the following variant of the `best_match` algorithm seen during the course:
  1. the score of a document must depend on the *frequency* of a query term in that document, where the frequency is the ratio between the number of occurrences of the term and total number of words in the document;
  2. returns only the 20 documents with higher score;
- the following optimization of `best_match`:
  1. sort documents in each inverted index in order of frequency of the term at which the inverted index refers (it can be done offline);
  2. for every query term define its possible *impact* on the score as the frequency of the most frequent document in its index;
  3. sort the query terms in decreasing order of impact;
  4. consider the first 20 documents in the index of the first query term (if the first query term has an index with less than 20 documents, then complete the list of 20 documents with the first documents in the index of the next query term);
  5. compute the score for each of these documents;
  6. consider the first term in which there are documents that have not been scored;
  7. consider the first non-scored document in the index of this term;
  8. if the frequency of the current term in the current document plus the sum of the impact of next terms is larger than the score of the 20-th scored document, then score this document and repeat from 7, otherwise consider the next term and repeat from 7;
  9. return the 20 documents with higher score.

For the ranking algorithm, the company also has two alternatives:

- PageRank (seen during the course);
- SpamMass (see Leskovec, Section 5.4.5).

You are required to implement these algorithms and to run experiments for evaluating which is the best choice for these algorithms. The experiment must be run on a set of approximatively 30000 pages: in particular, choose a web-page for each of the 15 categories listed in `https://www.dmoz.org/`, and for every of these web pages crawl 2000 pages by using the `Wibbi` online crawler.

Moreover, from each pair of sets of 2000 pages, choose at random 10 pairs of vertices (u, v) with u being a page in the first set and v being a page in the second set and add a link from u to v (if this link was absent).

Moreover, create a spam farm (see Leskovec, Section 5.4.1) with 100 pages plus one target page. The target page will contain the 500 words with the smaller index among the one contained in the above 3000 pages. Randomly choose 30 documents among the 30000 described above and add a link from them to the target page.

Execute the same queries (possibly consisting of more than one term) for each possible combination of ranking-matching algorithms. Compare the outputs in terms of accuracy and response time.

For the sponsored search, the company has to choose among two alternative auction formats.

- Budgeted First Price Auction with the `Balance` algorithm (seen during the course);
- Budgeted VCG with `Balance` algorithm (winners are selected according to the Balance algorithm, but they pay the harm they make to the *live* bidders, that is bidders whose current budget is at least the current bid)

You are required to implement these alternatives. Moreover, you are required to implement an *efficient* advertiser bot for each of the above auction format.

Here, a bot is said efficient if it gives high utility to the advertiser that uses it. Possible bot are the following:

- Best-response bot with balanced tie-breaking rule (seen during the course. Be careful, advertisers must best-respond to the last auction for that query!)
- Best-response bot with competitor-busting tie-breaking rule (submits the highest possible bid that gives the desired slot)
- Best-response bot with altruistic bidding tie-breaking rule (submits the lowest possible bid that gives the desired slot)
- Competitor-bursting bot (always submits a bid greater than the highest bid seen in previous auction, even if it is greater than own value)
- Budget-saving bot (always submits that is the minimum among the last non-winning bid and the advertiser value for the query)
- Random bot (always submits a random bid)
- Combination of the above bots based on the current budget or the advertiser value for the current query (e.g., *do competitor-bursting as long as your current budget is half the initial budget and then do best-response* or *do competitor bursting for queries for which the advertiser value is high and budget-saving for the others*)

You are required to implement these bot (and, possibly, every bot you think can be successful) and to make experiments to select which one is the most successful one (given the current auction format and given that all the other advertisers are adopting the same both).

In particular, for each auction format and for each bot adopted by the other advertiser, you must run the auction for at least 500 different auction settings (slots' click-through ratio, advertisers' values and budgets) randomly generated.

Finally, once the best bot for each auction format have been chosen, you must test which one gives the better revenue to your company (given that all advertisers use that bot).

BONUS: Can you compare naive implementations with parallel implementation of the algorithms?