Data Science in Production Exam

Spring 2022

Johan Ødum Lundberg

IT University of Copenhagen
jolu@itu.dk

June 3, 2022

Q1. List three possible ways of sorting the postings lists in an inverted index, and briefly discuss at least one advantage and one disadvantage of the different sorting strategies.

A1. Three possible ways are: Sorting by id, sorting by term frequency (tf) and sorting by PageRank.
Sorting by id only requires one accumulator, since the ids are sorted. Meaning, we can compute tf-idf, before reaching the next term. However, this is expensive since we can't stop early if we want to retrieve documents with tf $>$ k.
When sorting by tf, documents are no longer sorted by id, meaning we need to multiple accumulators when computing tf-idf. However, we can now stop early using some threshold, like tf $>$ k.
When sorting by PageRank,

Q2. Given a phrase query, how are search engines able to find relevant documents that contain that phrase query but with tokens that are not contiguous? For example, a document containing the sentence *"a black fat cat ate my homework"* is found relevant to the phrase query *"black cat"*. Please briefly describe the algorithms or datastructures that are relevant to this case.

A2. We will use an extension of the traditional inverted index which also contain a list of positional indexes. A search term is related to `N` number of documents that contain it and each of these contain a list of positional indexes where each index specify the term's position in the document. To return documents where the terms of the query possibly are non-contiguous, we add the constraint that neighbouring terms of the query must not be more than k distance apart.
Intermediate solutions could be indexing bigrams or trigrams etc however that is too space demanding.

Q3. The chief of search of an e-commerce service is tasked to evaluate the effectiveness of the website's search functionality. The marketing department has found that a good list of search result is one that includes at least one good relevant result towards the top of the list. One newly-hired data scientist in the team suggests that a good metric to evaluate search performance could be precision, calculated as the fraction of documents retrieved that are deemed relevant. Do you agree with the suggestion and why? Would you suggest to change or improve the evaluation metric?

A3. Yes, the marketing department wants a list of search results containing at least one relevant result, but the ordering does not matter, making this an unordered and binary search result. Precision@K is especially good here, since we might not know the set of all the relevant documents, and I would perhaps use that rather than the normal precision. However, if the e-commerce service do know the number of relevant results in the corpus, than perhaps R-relevance is better. This also fits the binary, unranked use case.

Q4. A study panel with users of a search engine for legal documents reveals that users find that on average the set of returned results for a query is too small. Interviews revealed that users often search for fairly technical terms that are found in only a handful of documents, but they would like to have an overview of documents that are topically related to their search intent, even if they don't contain the query. How would you change the search engine to better meet the users' needs? Explain briefly the technique you would use.

A4. We can use Rocchio's algorithm to retrieve all relevant documents of our initial query, get the tf-idf

vectors of all the terms in the top K documents, sum them and get the top M terms from the resulting vector. We can then submit a new query containing the initial terms and the new set of top-M terms.

Q5. A company wants to build a new search engine for patents to compete with Google Scholar. They have invested in a crawling infrastructure to download the title, abstract, and main body of millions of patents across the Web. The chief of staff does not know which of the three fields of text should be the best to be indexed for search queries. What would you suggest? Could you briefly describe how you would structure your suggested search strategy in the production search engine?

A5. I would suggest that we keep an index for each of the three fields and assign an importance weight to each of them. We should then learn the value of the weights using a supervised approach. We sum up the importance weights for each of the zones matched the given query.

Q6. Your manager has just seen a Youtube tutorial about Learning To Rank (LTR) and he is so excited about it that he wants you to implement it for the company's Web search engine (an engine that indexes all types of web pages, very much like Google does). He proposes to implement the following workflow. For each query, LTR should be used first to rank all possible results. Then, to improve the quality of top 10 results, a re-ranking based on the documents PageRank should be performed. Do you agree with the manager and why? What improvements would you possibly propose, if any?

A6. I would propose that we first use a native ranking method, BM25, to discard documents that are not relevant to handle recall. Then we should use a LTR model to re-rank matches by relevance, i.e. handle precision.

Q7. You work in the engineering department of an online forum. In the forum, user posts can be voted by other users on a scale from 1 (bad posts) to 5 (good posts). Management wants to implement a new feature to show "hot" posts, namely the most interesting posts published recently. A colleague of yours proposes that a simple but effective baseline to implement the service could be to show the set of posts with highest average score among the posts published in the last 24 hours. Is it a good idea and why? Would you propose any improvement to it? Please briefly describe your alternative solution, if any.

A7. I would not use the suggested idea, as it only considers posts published in the last 24 hours.
To avoid the ranking of a new post with very few ratings to skyrocket, we rank it by the lower bound of its 95% confidence interval. To account for the recency of a post, we discount the ranking of each post by its age to the power of some alpha, which we might then multiply by some factor F, depending on the type of post.

Q8. Explain the concept of monotonicity in the task of frequent itemset mining, and state why it is important for algorithms that extract frequent itemsets.

A8. If an itemset is frequent, all of its subsets must also be frequent. If one of the subsets is not frequent, then the itemset cannot be frequent. We can use this rule to prune out non-frequent items from itemsets until we are left with all the itemsets that satisfy our minimum support, for instance, itemsets that at least occur twice. Using the rule of minimum support, we can construct frequent item counts, frequent pairs counts, frequent triplets counts and so on until we are left with an empty set, meaning we have constructed all the frequent itemsets.

Q9. A new movie recommendation service has implemented a very sophisticated system of catalog creation. As a result, the service database contains virtually all movies ever shot and it is continuously updated with new movies. All movies are rich of metadata (actors, genres, box office performance, etc.). The service is still struggling to attract users, and at the moment only a very small set of hardcore, very active users is using it. Typically, these users get recommendations and rate the movies they have been recommended after they watch them. The management suspects that the site is not doing so well because the recommendations are given at random, and wonders which recommender systems should be implemented to keep existing users engaged, and to provide meaningful recommendations to the few new users who land on the platform for the first time. What solution(s) would you recommend to implement and why?

A9. I would recommend a content-based nearest neighbor approach. We first extract movie profiles,

index them into LSH and aggregate the movie profiles, those that user have rated, into a user profile. We can then query the LSH index with the user profile to get movie recommendations (candidate movies), ranked by cosine similarity. We then create another index containing the movies that the user have watched. We query this index with each recommended movie to predict a user's rating of each new movie. The predicted rating is based on the average of the ratings of k-nearest movies watched by the user.

The approach is better than a user-to-user approach, as it encodes information about what a user like about a movie. The user-to-user, on the other hand, compares the vectors of two users and recommends new movies if you seem to like the same as another user, which requires data on other users, the content-based approach does not. This approach would likely give more meaningful recommendations which could also be explained better, i.e. you will like this because it is similar to this. The approach will accommodate for unique tastes. Perhaps there are also unrated movies, as they have very few users, but this approach handles that too (cold start problem).

Q10. Describe a method to find efficiently the nearest neighbors of a given user in a user-user collaborative filtering system. Contextualize your description in a scenario where all users and items can be mapped to a n-dimensional latent space.

A10. For any two users, subtract their average rating from their rating vectors and compute their cosine similarity. These are each of length n. For a given user, rank other users by the cosine similarity. Run KNN to get top K nearest neighbors.

However, a more efficient approach would be to use LSH, such that, for a given user, the time to get a set of candidate similar users is near constant.

Q11. Given an $m \times n$ rating matrix R, with m users (rows) and n items (columns), explain what left/right singular vectors and singular values of the SVD decomposition of R represent.

A11. The left singular vectors of R contain information about the column space of R and is sorted in decreasing ability of explaining the variance in user ratings across items. A user spans the rows, i.e. every singular vector of the matrix U. Right singular vectors of R contain information about the row space of R, sorted in decreasing ability of explaining the variance of the rows, i.e. the variance in user ratings across users. An item spans a column (all the singular vectors) in the matrix V^T. The singular values encode the importance of the individual singular vectors (left and right).

We can recompute a representation of our matrix R using $\sigma_1 u_1 {v_1}^{*T}$, i.e. the most important singular vectors and value, since this explain most of the variance of R. If we want a better or more precise representation of R, we add some of the following important singular values and vectors.

Q12. Provide a high-level description (i.e., skipping implementation details) of a social link recommender system based on a local topology feature of your choice. Imagine a scenario of undirected, unweighted links (e.g., Facebook friendship). Describe how you would conduct an offline evaluation of that system.

A12. We will recommend new facebook friends to a user using common neighbours between the given user and other users. For evaluation, we will hide some of the known connections and try to predict them using leave one out cross validation.

Q15. Your company wants to advertise its product online in the banner of a popular website. The marketing chief doesn't know much about online ads, so it puts you on the phone with the representative of an advertising marketplace. The person on the phone offers you two different deals. The first deal follows a cost-per-time billing model with a priced-floored, first-price auction. The second deal follows a cost-per-click model with a second-price auction. Which deal do you pick and why?

A15. I pick the second deal as a cost-per-click model is easier to measure. We also only pay when a user actually clicks on our add. We can use CTR for this. The second model also follows a second-price type, which brings the total price down a bit, which is good for us, the advertiser.

Q16. Your company is at an historical crossroad. The CEO decided to change the color of the logo for good. The three top color options from the design department are pink,

lime, and turquoise, and the data science department is asked to find which one is most effective in attracting website visits. One day, you find two chiefs of data science fighting in the coffee room over this issue. One believes that a bandit approach should be used to determine the best color. The other believes that A/B testing should be used instead. Who do you think is right and why?

A16. Since the colour change is a long-lasting design change, we should use A/B testing. If we needed a decision quickly, in order to optimize the click through rate, we could use a bandit approach, however this is not the case. In this case, it is not a problem that A/B spends more time on sub-optimal choices, it however reaches statistical significance faster than the bandit approach.

Q19. Your roommate has finished an online Docker course and now they are all hyped-up. They want to build a dockerized version of their own webpage. They plan to write a Dockerfile to define a container. To enhance performance, they will skip the OS layer and have a bare-metal installation of Flask as first layer. As a second layer, they will have the latest Python version, to ensure that Flask can run properly. They will then declare a volume to open the web application to an internal IP and port, so that anyone can access the website from the World Wide Web. Once the container is defined, they plan to instantiate it in an image on a cloud service, so that the website becomes live. Please explain what's wrong with your roommate's plan.

A19. First of all, to open the webserver up to the world does not require declaring a volume. Rather you should expose the webserver to a specific port, which opens service on the container.

Next, it sounds like he needs multiple containers, one for the webserver, one for volume, and one to connect them. Thus he should also have mentioned a YAML file to specify how these should be composed and interact.

Lastly, the OS layer in docker cannot be avoided. It is required due to the shared kernel requirement. However, you don't necessarily have to specify it in your Dockerfile, but it is always the bottom layer of an image. Also, the second layer of the dockerfile should be the Python version and the third layer, the Flask module, as this is dependent on Python.

Q20. What is the concept of "Desired state management" in Kubernetes (k8s) and what are the components that help achieving it?

A20. This revolves around the master node trying to bridge the gap between the state of the system and the desired state of the system. What happens is: First the Kube controller manager detects a change in the kubernetes cluster. It compares the desired state with the actual state and tries to recover the desired state by triggering an action through the Scheduler. The actual state is retrieved from Etcd which is used for storage of key-value pairs that describe the state of the cluster. The state is stored/changed by instruction from the API of the master node, which also tells the controller to start monitoring the new resource (once deployed). The API also tells the scheduler to deploy the new resource on the worker nodes. The scheduler then uses the API to schedule pods on workers by communicating with the Kubelet on every node. The Kubelet of every node then reports back to the API of the master node. The API, scheduler, Kube controller manager and Etcd are all elements of a master node.
In the case of worker node failing, the scheduler is asked by the controller to reallocate any failing pods on an available worker node, which fixes the state. The cluster configuration is specified (what we want) in a YAML file, Kubernetes monitors the cluster and tries to bridge the difference between the YAML file and the actual state of the cluster.