CS 549 – Research Paper Summaries

Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS

Students: YuWen Chen & Rahul Damineni

- What is the problem discussed in the paper?
  - Markov Logic Networks are slow to train and infer on.
  - The inference consists of grounding and search phases.
  - The paper proposes a bottom-up approach to construct SAT formula that leverages RDBMS optimizer
  - They propose a parallelizable solution to search weights for the above formula – thereby speeding up inference
- Why is it important?
  - MLNs play crucial role in building Q&A systems.
  - It can be built leveraging vast amounts of data available on the internet, but it has to be fast at least during inference to support querying
- What are the main ideas of the proposed solution for the problem?
  - To improve performance of grounding phase – a bottom-up approach that leverages DBMS optimizer was suggested for formula construction
  - Uses data partitioning techniques to increase parallelism by searching solutions for connected components of the derived formula.
- What are the strengths and weaknesses of the approach proposed in the paper?
  - Partition phenomena generalized to arbitrary MLN
  - Space efficient

Actions

Deep Learning for Entity Matching: A Design Space Exploration

- What is the problem discussed in the paper?
  - Given a couple of relations with the exact same schema, how can you match entities between these tables?
  - These relations can be structured data with short and atomic attributes; dirty data where attributes aren't accurately segregated under their columns all the time; and textual.
  - Especially, how would you solve the above problems if you can't just expect the input pair strings are semantically similar but not syntactically so.
- Why is it important?
  - We can automatically integrate two databases.
- What are the main ideas of the proposed solution for the problem?
  - This matching was set-up as a binary classification task: a deep neural network will be trained end-to-end to encode and summarize tuples into a distributed representation (high-level embedding) and learn to classify such representation as a "match" or a "no-match".
  - As a part of exploring and optimizing various components of this solution:
    - Bi-directional GRU was used to encode each attribute of the input pair so that sequential information of their tokens are preserved.
    - Attention-based encoding was used to soft-align the tokens (by considering both attributes jointly) of each attribute – this brought representations of "semantically similar but jumbled adjectives of entities" closer
    - A hybrid method that sequentially embeds and then soft-aligns was proposed to capture goodness from the above two methods
- What are the strengths and weaknesses of the approach proposed in the paper?
  - Many components of design space were thoroughly studied and conclusions are published. The following interesting questions are answered:
    - When will character-based attribute embedding work? And when should you learn this from scratch?
    - How would you choose model complexity based on dataset size? How would you minimize training time?
    - How would you choose comparison distance metric b/w fixed and learned ones depending on the kind of representations you'd at the classification layer?
- How to extend this work?
  - Currently, the knowledge retained when we train the proposed model does not readily work for a new dataset. The architecture doesn't really place and restrictions for such a transfer. It accepts an arbitrary number of attributes as input (as long as the number of attributes in the input pair matches). As the authors suggest, incorporating the canonical knowledge base as a part of the model and matching attributes of each (arbitrary) input tuple to a node in this KB and to construct a combined representation on which classifier would be trained is the way forward. Word embedding partly does this – but our representation should form a signature based on attribute combination to represent an entity uniquely. The central idea is – if we know all properties of the world, we can describe any entity uniquely.

Stochastic Database Cracking

- What is the problem discussed in the paper?
  - How column-store cracking helps in answering sequential workloads without having to sort the whole column?
- Why is it important?
  - Sorting column stores could be extremely expensive in terms of time/ resources.
  - A lazy approach like cracking based on workload is a good tradeoff b/w scanning and sorting.
- What are the main ideas of the proposed solution for the problem?
  - Cracking on random workload is naturally effective but usually sequential workloads are popular.
  - So a sense of randomization was introduced by cracking the store randomly and then cracking based on bounds.
  - Median based cracking is also an effective idea to quickly narrow down to the required bound but finding median is a costly operation.
- What are the strengths and weaknesses of the approach proposed in the paper?
  - Linear space and logarithmic time algorithm

Learning to Sample

Students: YuWen Chen & Rahul Damineni.

- What is the problem discussed in the paper?
  - How to efficiently get the count of objects in a relation that satisfies a complex predicate (q)?
- Why is it important?
  - OLAP systems are mostly used to get quick estimates rather highly precise estimate that would take enormous time and resources
  - The two important factors for getting such estimates are time/ resources spent in obtaining it and the knowledge of the estimate's accuracy.
- What are the main ideas of the proposed solution for the problem?
  - There are sampling-based methods that could get the estimates but evaluating complicated predicate on a random sample of a good size to achieve the required confidence interval is costly
  - There are ways to approximate predicate function using Machine Learning and then use an SRS to get an estimate – but we won't be able to judge the confidence of such estimates since ML classifiers are trained on a sample
  - But if we train an ML classifier to derive a well-stratified sample whose objects are cheaply classified using the classifier, we are making use of the best of both worlds. We will have confidence intervals from the Stratified sample and evaluate objects cheaply using a trained classifier. This is termed as Learned Stratified Sampling.
- What are the strengths and weaknesses of the approach proposed in the paper?
  - Strengths are to quickly converge to a well-stratified sample
- (extra) How can one extend this work?
  - Maybe train the classifier and LSS back and forth to improve convergence time?

MULTI-AGENT COOPERATION AND THE EMERGENCE OF (NATURAL) LANGUAGE

Students: YuWen Chen & Rahul Damineni.

- What is the problem discussed in the paper?
  - How can two agents interact with each other and learn to communicate from scratch? Evolving a new personal language
  - How can we align their personal language to meet our natural languages?
- Why is it important?
  - If agents can communicate, we can break bigger problems into smaller ones and let each agent specialize in solving a subproblem. Then we can link a couple of agents to achieve a comprehensive task by communicating with one another.
  - If we (humans) can interpret the agent's communication, then we can have our agent do some "thinking" for us and learn outcomes of though chains – a good win for AGI.
- What are the main ideas of the proposed solution for the problem?
  - To help communicate: the sender's embedding is aligned with the receiver's by applying the reward. The reward forces sender to stabilize and have a fixed policy for choosing symbols for a given image.
  - To help ground the language to natural language: they are constraining "what" symbol to assign to which semantic (image) by extending the sender to perform supervised labeling task.
- What are the strengths and weaknesses of the approach proposed in the paper?
  - Great idea, working at a root level of NLU!
  - See my second point on extending this work that could be a possible weakness.
- How to extend this work?
  - The most interesting thing is, the neural networks are capturing metonymic intuitions with-in the languages.
  - For that to happen, our language should have some intuitive similarity b/w words (which it does, because most words are derived from the root words).
  - First, I'd try to come up with gibberish language which doesn't feel logical (humans should find it harder to learn and impossible to extend) and then see if the grounding to NL results vanishes now. If not, that would be extremely interesting.
  - Second, to improve grounding, I'd use distributed word representations in place of symbols (from the sender) so the receiver has more expressive power. We can generalize this to speech as well, which has even more expressive power. The downside is the increased game count for convergence.

The Data Interaction Game

Students: YuWen Chen & Rahul Damineni.

- What is the problem discussed in the paper?
  - A user would have an intent in their mind while "querying" a DBMS.
  - They formulate this query without knowing the DBMS schema or architecture so it'll be vague (can inform multiple intents)
  - A DBMS' job is to map this query to intent and return the results.
  - If this intent matches user intent, then DBMS did its job.
  - During this interaction, based on responses returned by DBMS for their queries, user strategies, and changes their query forming style (keeping intent consistent throughout)
- Why is it important?
  - **Existing DBMS doesn't consider that users adapt while querying based on the responses they get and lose out on an important feedback signal**
- What are the main ideas of the proposed solution for the problem?
  - The back and forth interaction b/w DBMS and the user can be modeled as a game whose aim is to maximize collective reward (user is happy when they see relevant stuff, DBMS is happy when it gets user's intent right)
  - To facilitate exploration while not boring user with completely irrelevant content, a weighted ranking of tuples based on their relevance is done
  - To propagate reward and not materialize candidate networks (join tables) Poisson Olken algo uses sampling
- What are the strengths and weaknesses of the approach proposed in the paper?
  - Interaction game model captures both sides of the coin by unifying their reward
    - Effectively facilitates exploration using a weighted ranking algorithm
    - Uses sampling of base tables to produce joins

Ripple Join For Online Aggregation

Actions

- What is the problem discussed in the paper?
  - How do you perform aggregations (like sum, avg, count, etc) on the result of a join in an online fashion – meaning, displaying the estimates with confidence intervals instead of blocking the output until the query terminates?
- Why is it important?
  - To get the big picture of a dataset quickly – independent of the size of the input relations
- What are the main ideas of the proposed solution for the problem?
  - We make use of sampling to construct a subset of join relation and the compute aggregations on that.
  - Using CLT to provide confidence intervals on estimates
  - Sampling one relation more than the other (rectangle join over the square join) improves the confidence interval
  -

The PageRank Citation Ranking: Bringing Order to the Web

Actions

- What is the problem discussed in the paper?
  - How do you generally rank a set of pages w.r.t each other based on human interest and attention devoted to them?
- Why is it important?
  - The importance of a page could be actively determined by its content. But comprehending that is a hard problem. Which is why we chose passive feedback: how much do humans care about this page?
  - Once we develop a good method to rank pages based on such an approach, we can order user query responses in decreasing order of their usefulness.
- What are the main ideas of the proposed solution for the problem?
  - Peer-feedback.
  - We will measure how many backlinks we discover for a given page
  - We will weight each backlink based on who the referrer is.
  - The rank of a page is sum of weights of each backlink.
  - The important idea is, we will conserve the total energy (rank?) of all webpages combined.
  - We do so by letting each page lose some energy when it refers to another page.
- What are the strengths and weaknesses of the approach proposed in the paper?
  - [TODO]
- (extra) How can one extend this work?
  - Maybe initialize the input source for each page based on its content.
  - Hand label a few good pages for each topic and use them to judge "how good" a given page is.
  - The better the page, the higher its initial rank is.

Authoritative Sources in a Hyperlinked Environment

Actions

- What is the problem discussed in the paper?
  - Given an interconnected web of pages and a query "q", rank top k authoritative pages.
  - Given a page "p", rank top k similar pages.
- Why is it important?
  - Effective information retrieval for the user. The user pose queries with the intent of learning the relevant information. Although the database may contain the right information among a bunch of irrelevant information, if the system can't pinpoint the exact answer the user is looking for, he/she has to put that effort. With the volume of information we have, it may be impossible for anyone to find the right information from a pile.
- What are the main ideas of the proposed solution for the problem?
  - Instead of using textual information from pages, the paper discusses harvesting relevance information from page interconnections (links)
  - The main idea is, given a set of possibly relevant pages for a query, an authoritative page would be the one that's highly referenced to.
    - However, some generally popular pages (like ads) are highly referenced irrespective of the query.
    - To solve this, the authors recognized "hub" pages. The pages from which authorities are "referenced from"
    - If it's a weak hub, it won't reference many authorities for this particular query.
    - A fake authority would be the one with a large in-degree but smaller average hub score.
    - In this sense, hub and authority scores reinforce each other iteratively by counting and normalizing in and out degrees.
    - In the end, we can display top k queries based on magnitudes of authority scores
- What are the strengths and weaknesses of the approach proposed in the paper?
  - The solution depends on the initial subset of pages that would be used to pick pages that are most referenced in and out of this subset. Although this seems like the only way, this is introducing a bias to the final subgraph.
- (extra) How can one extend this work?
  - I would change the way the subgraph is constructed before applying the iterative algorithm
    - I would make use of textual information from pages to narrow down in the initial subset from which I'll pick in & out referenced pages.
    - This could be computationally intensive, which is why the author did what they did.

Presentation

- Why not manually write the matcher?
  - Other options are:
    - Rule based

- Crowdsourcing
- Human experts to help refine the learned matching functions

- What is the point in learning the matcher? What is the guarantee that it will generalise?
- Prime differences b/w different EM tasks
- Entity matching = Blocking + Matching
- Different ways to setup the learner:
  - Binary classification problem
  - Triplet learning —> embeddings —> NN classifier
  - Learning entity embeddings using contrastive loss —> NN classier
  - Nearest neighbour search
  - Matching networks
- Show pictures of DL layers
- How is soft alignment matrix W computed?
- What/ where are learnable parameters in the models?
-

Model training

- GRU
  - Hidden dim = 50 seem to be overfitting (v0)
    - Reduced hidden dim = 20, still overfitting (v0.1)
  - Increased lr to 0.01 from 0.0001, doesn't seem to affect (v0.2)
  - Let's perform overfitting test (v0.3)
    - Passed! Tried various versions of v0.3 to figure out hidden dims & learning rate
  - Let's understand how distribution skew affects, more positives should speed up training(v0.4)
    - This turned out a little weird. Let's skip it for now.
  - Training on a large dataset with params learned so far – shit bricks! (v0.5)
    - It's overfitting but isn't actually learning much. Then I inspected data and made a simpler dataset

- After cleaning comments, now I will just train with two different authors (v0.6)
  - It’s the easiest thing that model could learn, if at all.
  - Reducing learning rate and model complexity may have been working! After all, all along, the training data wasn’t fitting too! May be the patterns as too sparse?

gru_v0.6_two_auth_1k_h25_l0.0001_crossentropy_loss

- Finally, BCE loss + ordered example mixing helped! (v0.6)
- Using v0.6, let’s now increase data volume (full dataset, v0.6 params, ordered mixing) (v0.7)
  - Got a personal best validation accuracy of 84%; hidden dim = 150, 2k samples
- Using v0.7, further tune hyper params (v0.8)

CS 519 [Applied Machine Learning]
Paper Review: XGBoost

- Regularisation of trees Explicit VS Implicit
  - Trading regularisation with predictive power of the model
- Sparse data automated imputation with unified data model
- Determining splitting points without brute forcing
  - Candidate splitting points: maximise the spread, quantile sketching offers ready to go solution
  - Cache aware prefetching
- Out of core compution
  - Distributed computing