Search engine for generating a movie cast

My project subtitle

Christoffer Holmesland
University of Stavanger, Norway
cr.holmesland@stud.uis.no

Malavika Ramakrishnan
University of Stavanger, Norway
m.ramakrishnan@stud.uis.no

ABSTRACT

This document serves both as a template and a guideline for your report. You can replace headings and body text with your content. Before you start writing, however, we recommend that you carefully consider the instructions in this document. In this paper we show how we used data from IMDB(source) to generate a movie cast based on user input. To determine the recommended cast we calculate a score based on genres, actor to actor relationship, and a brief movie summary. The user is presented with a list of actors and the predicted IMDB user rating score.

KEYWORDS

Haddop, Spark, Film, Casting

1 INTRODUCTION

Context and Motivation. For a low budget movie it is important to have a well-known actor. It can increase the box office sales or help attract investors (source stephenfollows). If you are new to the film industry it can be hard to know which actors or actresses will be a good fit for your movie. It is possible to use previous movies where you liked the performance of the actor as inspiration, but there are too many movies being made that it is not possible to get a proper overview of all the possibilities. Another possibility is to look at websites like IMDB (source) to find inspiration. You can use features like the user score of a movie to select actors. We do not consider this as a good option because there are too many movies to compare, and in practice you will end up looking at only the top rated movies.

Research Problem. In this paper we present a solution to the problem of selecting a movie cast. Our system lets the user input a brief summary of their movie and a list of actor characteristics they want the lead actors to have. The system uses data from IMDB to rank possible actors and returns a list of suggestions to the user. The rank is based on a combination of past performance, acting relationships and a subset of the actor characteristics.

Contribution Summary.

- We implement Paxos using brand new technologies.

2 RELATED WORK

The IMDB datasets have been used to predict the gross movie revenue (source jae huang). The accuracy of several machine learning models were compared to find the best way of predicting the gross revenue. The results show that the random forest model had the best performance and that the most important feature is the number of user votes. The datasets have also been used to compare the user rating with other features (source max woolf). In the same analysis they found that lead actors are typically ten years older than lead actresses.

The movie project system cinelytic developed by Cinelytic, Inc. (source) is able to analyze and produce forecasts of how well a movie will perform based on country, cast and distributor. The system ranks actors based on the predicted economic increase they have for a given project.

Our system is similar to cinelytic, but we are basing the score only on IMDB data and use a subset of the features considered by cinelytic.

3 BACKGROUND

Give a short, self-contained summary of necessary background information.

For our example if your project is a new Paxos implementation, the background would be a brief introduction to the consensus problem, the Paxos algorithm, the system model used by the Paxos algorithm and common optimizations and relevant variations of that algorithm.

The goal of the background section is to make the paper self-contained for an audience as large as possible. As in every section you start with a very brief overview of the section. For our example that would be: In this section, we introduce the consensus problem, the Paxos algorithm and common optimizations of that algorithm.

If you copy a definition from a text book or some other source, you must cite that source. See Example 8.1 below.

4 HADOOP CLUSTER

Our Hadoop cluster consists of one name node and three slave nodes. The name node has 8 GB RAM and a 2.4 GHz Intel Skylake processor with 4 cores. The three slave nodes have 4 GB RAM each and a 2.4 GHz Intel Skylake processor with 2 cores. We used Hadoop version 3.2.1, Spark version 3.0.0-preview2 and Python version 3.5.2.

5 YOUR METHOD

5.1 Design

There are three parts to the system. First we had to get all of the required data. Most of it can be downloaded from the IMDB website(source to imdb/interfaces). This data does not include actor gender, movie plot or box office values. To determine the gender we used the list of known professions for every person. If they are known for being a actor they are assigned the label "0". Actresses

Supervised by Leander Jehl and Hein Meling.

Project in Data-intensive Systems (DAT500), IDE, UiS
2020.

are assigned the label "1". Some people in the dataset have never acted, for example directors or producers. They are removed from the data. There is no way to find the movie plot in the datasets, so we had to find an external source. One alternative is to use webscraping on the IMDB website. We tried doing this but quickly realized that it would not be possible for us because they stop responding to requests after too many in a given time period. We were able to get about 9 000 requests per hour. Since the dataset includes about 500 000 movies this would not be a good solution. It was also not possible for us to know if there were other rate limits e.g. 50 000 requests per day. Instead we used the OMDB API (source) to retrieve the movie plots. Using this API we were able to get all of the plots in 1 hour and 43 minutes. Our original idea was to also make a prediction on the box office value. After using the OMDB API to also find the box office values we saw that only 6381 movies have this value. We do not believe this is enough data to make proper predictions on (why?) so we did not use this data. Instead we decided to generate a prediction for the IMDB user score because this value is available for all of the movies in our dataset.

Concisely present your design. Focus on novel aspects, but avoid implementation details. Use pseudo-code and figures to better explain your design, but also present and explain these in the text. To assist in evaluating your design choices, it may be relevant to describe several distinct *design alternatives* that can later be compared.

5.2 Analysis

5.3 Optimization

Explain how you optimized your design and adjusted it to specific situations. Generally, as important as the final results is to show that you took a structured, organized approach to the optimization and that you explain why you did what you did. Be careful to argue, why your optimization does not break the correctness of your design, established above. It is often a good strategy to explain a design or protocol in stepwise refinements, so as to more easily convince the reader of its correctness.

5.4 Implementation

It is not necessary to "explain" your code. However, in some cases it may be relevant to highlight additional contributions given by your implementation. Examples for such contributions are:

- *Abstractions and modules*: If your implementation is nicely separated into interacting modules with separated responsibilities you could explain this structure, and why it is good/better than some other alternative structure.
- *Optimization*: If you spend significant time optimizing your code, e.g. using profiling tools, the result of such optimization may be presented as a contribution. In this case, reason, why the optimized code works better.
- *Evaluation framework*: If you implemented a framework or application to evaluate your implementation against existing work, or in a specific scenario, this framework may be presented as a contribution.

Make sure to cite all external resources used.

6 EXPERIMENTAL EVALUATION

Here you evaluate your work using experiments. You start again with a very short summary of the section. The typical structure follows.

6.1 Experimental setup

Specify the context and setup of your experiments. This includes e.g. what hardware (VMs) you are running on, what operating system these machines are running, how they are connected, ...

Also explain how you generate load for your system and what parameters you used here. The general idea is to include enough information for others to reproduce your experiments. To that end, you should provide a detailed set of instructions for repeating your experiments. These instructions should not be included in the report, but should be provided as part of the source code repository on GitHub, typically as the `README.md` file, or as Shell scripts or Ansible scripts.

If your experiments give strange or unexpected results, analyze, profile and debug your code. **Do not simply re-run experiments until they give the expected results.**

Finally, running experiments is very time consuming and you may need to go through multiple rounds of experiment, debugging and optimization. **Do not delay running experiments until the end of your project period.**

6.2 Results

The results of your experiments. Compare different variants of your design (e.g. with and without optimizations) or compare performance to other designs or systems. Plots should show the average over multiple runs (at least 10 as a rule of thumb), including error bars, percentiles or min/max values.

Discuss the plot and extract the overall performance. Do not repeat all numbers in the text, but mention relevant differences in numbers, e.g. our optimization improves throughput by 26%. Discuss how the results validate or contradict your assumptions.

Perform experiments to evaluate your system under operating normal conditions, when experiencing failures or attacks, or with different workloads.

Figure 1 shows a graph generated with `pgfplots` from experiment data.

7 CONCLUSION

Here you need to summarize what you did and why this is important. Do not take the abstract and put it in the past tense. Remember, now the reader has (hopefully) read the paper, so it is a very different situation from the abstract. Try to highlight important results and say the things you really want to get across such as high-level statements (e.g., we believe that .... is the right approach to .... Even though we only considered LAN, the .... technique should be applicable ....) You can also formulate next steps if you want. Be brief.

8 CITATION

It is important that you correctly refer to sources.

- If you describe background, according to some textbook, cite that textbook. *See Example 8.1 below.*

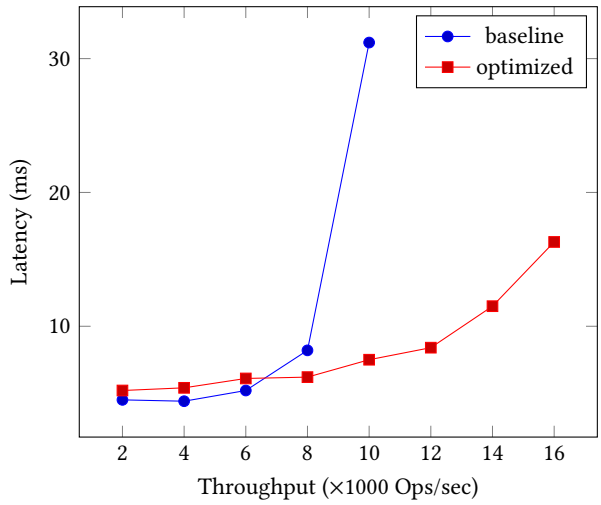

Figure 1: A graph showing latency and throughput of a baseline and optimized implementation. The axes show latency in milliseconds, and throughput in thousand operations per second. Data is made up.

- If you make a claim about trends in industry or research, add citations to validate the claim. *See Example 8.2 below.*
- **Do not copy paste text from any source into your report without citation. It will be considered plagiarism.**
- Always use the tilde character between the text and the cite command, e.g.:

  `DBLP~\cite{dplg}`

  The tilde inserts a space, but prevents line break between the text and citation.
- For correct citations we recommend that you export bibtex entries from DBLP [4]. But make sure to update the bibtex entries to adhere to the following rules.

Here are some rules to be followed for your bibliography:

- Make sure titles are correctly capitalized.
- Make sure there is enough info for a reader to find the document: authors plus title isn't enough.
- Don't blindly copy and paste ACM bibtex entries without fixing them:
  - they list "New York, NY" as the address field, which means that all ACM conferences look like they took place there; just delete this, or - better
  - replace it with where the conference took place, which is much more helpful to the reader than the publisher's address
  - they have a truly weird idea about how to capitalize conference names; just correct this (consistently)
  * IEEE is equally bad, and uses a partially-inverted form of conference names.
  * Please actually review the bibliography before submitting your paper!

And while you are at it, please:

- Add includemathpmtc in the main LaTeX file, so you don't have math stuff in a different font than the main text.
- Always use the tilde character between the number and unit, e.g. 100 Mbps or 53 ms. The tilde inserts a space, but prevents line break between the number and unit.
- Never put SI units in italics.
- Do differentiate between bits (b) and bytes (B), and between powers of 10 (MB) and 2 (MiB).
- Avoid things like: "We refer the reader to [42]." That is, don't use citations as nouns.

Example 8.1. Consensus is a distributed programming abstraction that fulfills the following properties [2]:

Termination
Validity
Integrity
Agreement

Example 8.2. Consensus systems build a fundamental part of todays cloud-computing infrastructure [1, 3, 5, 6].

9 LATEX

This section contains some instructions and examples for using LaTeX.

9.1 You can add subsections

You can use numbered subsections to structure your sections or `\paragraph` to separate paragraphs with a heading using only a few words, as used in Section 1.

(1) This is an item in an enumeration.
   - This is an item of a unnumbered list. In this case, the lists are nested within each other.

(2) The second point in the enumerated list.

I can refer to the element of the enumeration above as Point 1. If you refer to numbered items, e.g. items from a list or figures or sections, always capitalize the name. For example, this is Section 9.

9.2 Figures

You can include figures. You can include files, as done in Figure 2. Avoid including jpeg, gif or bmp files since these do not scale nicely. Again Figure 2 is an example for that.

You can create graphs from your experiment-data using `pgfplots`. See the example in `tex/evaluation.tex` and documentation http://pgfplots.sourceforge.net/pgfplots.pdf.

9.3 Other tips

- Always use the tilde character between the number and unit, e.g. 100 Mbps or 53 ms. The tilde inserts a space, but prevents line break between the number and unit.
- Never put SI units in italics.
- Do differentiate between bits (b) and bytes (B), and between powers of 10 (MB) and 2 (MiB).
- Avoid things like: "We refer the reader to [42]." That is, don't use citations as nouns.

For further instructions on how to add **Tables, Algorithms, Theorems see acmguide.pdf**.

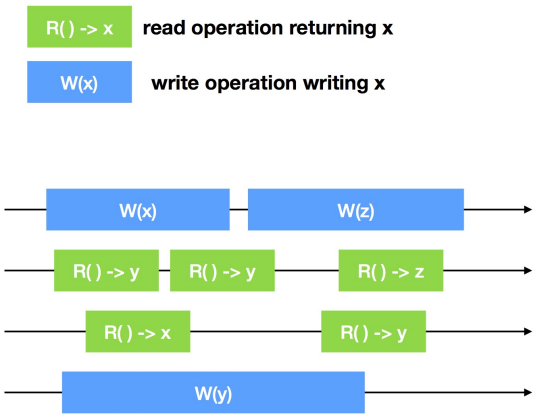

Figure 2: Figure taken from [7]

10 FURTHER READING

Here are some other useful resources: (make these into references instead of links?)

- The co-chairs of the Ninth SOSP prepared some recommendations and valuable lessons in a paper titled *How (and How Not) to Write a Good Systems Paper* https://www.usenix.org/legacy/publications/library/proceedings/dsl97/good_paper.html
- http://gramoli.redbellyblockchain.io/web/doc/talks/researchmethod.pdf
- https://www.doc.ic.ac.uk/~prp/doc/talks/11-prp-paper_writing.pdf
- This template is inspired by a template from Markus Püschel [8].

REFERENCES

[1] Michael Burrows. 2006. The Chubby Lock Service for Loosely-Coupled Distributed Systems. In *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*. 335–350. http://www.usenix.org/events/osdi06/tech/burrows.html

[2] Christian Cachin, Rachid Guerraoui, and Luís E. T. Rodrigues. 2011. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer. https://doi.org/10.1007/978-3-642-15260-3

[3] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson C. Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. 2013. Spanner: Google's Globally Distributed Database. *ACM Trans. Comput. Syst.* 31, 3 (2013), 8:1–8:22. https://doi.org/10.1145/2491245

[4] Schloss Dagstuhl and University of Trier. 2018. DPLG Computer Science Bibliography. http://dblp.uni-trier.de/

[5] Apache Software Foundation. 2010. ZooKeeper. https://zookeeper.apache.org/

[6] Patrick Hunt, Mahadev Konar, Flavio Paiva Junqueira, and Benjamin Reed. 2010. ZooKeeper: Wait-free Coordination for Internet-scale Systems. In *2010 USENIX Annual Technical Conference, Boston, MA, USA, June 23-25, 2010*. https://www.usenix.org/conference/usenix-atc-10/zookeeper-wait-free-coordination-internet-scale-systems

[7] Leander Jehl. 2018. Lecture Notes for DAT520 Distributed Systems.

[8] Markus Püschel. 2011. A Descriptive Title, not too general, not too long. https://www.inf.ethz.ch/personal/markusp/teaching/263-2300-ETH-spring11/project/report.pdf