Paper Categories

Traditional technical papers
Traditional technical papers present new solutions to well-defined problems. Expected sections include an introduction (including problem statement), a description of related work, a description of the new solution, and an evaluation of the new solution.

The acceptance criteria include: Is the solution novel? It is non-trivial? Is it useful? Is the solution clearly described? Is relevant existing work cited? Is the evaluation of the solution convincing?

Example traditional technical papers:

- Leslie Lamport, Time, clocks, and the ordering of events in a distributed system, 1978. link
- Tomas Mikolov et al, Efficient Estimation of Word Representations in Vector Space, 2013. link

Experimental papers
Experimental papers report on experiments with existing algorithms and tools. For example, experimental papers might compare the performance of several tools on benchmark data. Expected sections include an introduction, a description of the experimental setup and benchmark data, and experimental results.

The acceptance criteria include: Is the algorithm or tool important? Is there something different about the reported experiment from earlier experiments on the same algorithms/tools? Are the evaluation metrics meaningful? Was care taken in the design of the benchmark data set? Finally, do the experiments test for something of real interest, and are the results clearly reported through tables and plots?

Example experimental papers:

- Yiming Yang and Jan O. Pedersen, A Comparative Study on Feature Selection in Text Categorization, 1997. link
- Robert Ronngren and Rassul Ayani, A Comparative Study of Parallel and Sequential Priority Queue Algorithms, 1997. link

Implementation papers
Implementation papers describe the implementation or re-implementation of an algorithm. For example, such a paper might describe the implementation of a newly-developed algorithm, or might describe a new, improved implementation of an algorithm. Expected sections include an introduction, an overview of the algorithm, a description of the issues in implementing the algorithm, a summary of the new implementation, experimental results on the new implementation (and old ones, if any exist), and a summary of insights and lessons learned.

The acceptance criteria include: Is the algorithm important? Is there a significant issue with existing implementations that motivates another implementation? Is the idea behind the new implementation non-trivial? Are the experiments well-designed and are the experimental results reported clearly?

Example implementation papers:

- Ernesto Martins and Marta Pascoal, A new implementation of Yen's ranking loopless paths algorithm, 2002. link
- E. Hadjiconstantinou and N. Christofides, An efficient implementation of an algorithm for finding K shortest simple paths, 1999. link

Literature surveys

Literature surveys provide an up-to-date and well-structured overview of the literature in a specific area. One of the values of literature surveys is that research gaps are made explicit, which is helpful for readers entering a new research area. Additionally, reviews can outline the advantages and disadvantages of methods used in practice and discuss the implications of any findings. This is helpful for readers who need to interpret and use the findings.

Expected sections include an abstract, introduction, discussion, conclusion, and related work. The discussion section should list, describe and compare the leading work in the area. The conclusion section should summarize the survey by listing the technologies/methods that were discussed and compared.

The acceptance criteria include: is the survey comprehensive and up to date? Are sufficient details of cited papers provided? Is the survey easy to read and understand? Does the survey connect work in the field?

Example literature surveys:

- Douglas Kunda, Alinaswe Siame, A Systematic Literature Review, 2017. link
- M. Praveena, V. Jaiganesh, A Literature Review on Supervised Machine Learning Algorithms and Boosting Process, 2017. link
- Rini John, Nikita Palaskar, A Survey of Various Query Optimization Techniques, 2017. link

Tutorials

Tutorials attempt to make an algorithm or computational idea accessible by providing a clear presentation that minimizes the background knowledge expected of the reader. The sections expected in tutorials will depend on the nature of the topic being presented.

The acceptance criteria include: Is it well-written? Is the pedagogical approach good? It is technically sound? Does it improve upon other tutorials on the same subject (if available)?

Example tutorials:

- Jonathon Shlens, A Tutorial on Principal Component Analysis, 2014. link
- Lawrence Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, 1989. link

Instructional pearls
Instructional pearls explain a useful programming technique, approach, or design via straightforward and engaging examples. The topic of a pearl does not need to be novel, though it should not be familiar to a general CS audience.

Expected sections include an abstract, introduction, some number of body sections discussing the contribution, and a conclusion. A related work section is optional, though the introduction should provide some context behind the pearl. Body sections, and perhaps even the introduction itself, should be heavy on code, and many pearls incrementally construct pieces of a larger example.

The acceptance criteria include: how useful is this technique/approach/design? How new or unfamiliar is it? Is it relatively clean and elegant? Is the paper easy to read and understand? Are examples informative and complete? Is the paper engaging?

Some discussion of how to write pearls specific to functional programming are available here, and much of this transfers to instructional pearls in general. A large number of pearls in functional programming are available here; a selection of specific examples follows:

- Jeremy Gibbons and David Lester and Richard Bird, Enumerating the Rationals, 2006. link
- Martin Erwig and Steve Kollmansberger, Probabilistic functional programming in Haskell, 2006. link
- Ralf Hinze, A fresh look at binary search trees, 2002. link